MASTER EN SCIENCES INFORMATIQUES

Alignement business-IT: le cas de l'ERP Odoo

Patz, Michel

Award date:
2017

Awarding institution:
Universite de Namur

Université de Namur
Faculté d'informatique
Année académique 2016–2017

Alignement business-IT: le cas de l'ERP Odoo

Michel Patz

Promoteur : _____ (Signature pour approbation du dépôt - REE art. 40)
Michaël Petit

Co-promoteur : Benoît Vanderose

Mémoire présenté en vue de l'obtention du grade de
Master en Sciences Informatiques.

Résumé

Historiquement seules les grandes entreprises avaient les moyens de se lancer dans des projets couteux que sont les projets ERP. De plus, les projets ERP sont risqués car s'ils sont mal menés ils peuvent coûter assez cher. Dans le contexte des PME, celles-ci n'ont pas les moyens d'investir dans de gros projets et ne peuvent pas se permettre d'assumer les coûts de l'échec d'une implémentation d'un ERP.

Ce mémoire aborde donc la complexité de la mise en œuvre de l'ERP Odoo dans des PME. Nous aborderons aussi le thème de la variabilité avec les Feature model, et le thème de la mesure de la taille de modifications logiciel avec Cosmic

Notre objectif est d'apporter une méthodologie d'implémentation pour l'ERP Odoo simple et adaptée pour les PME. Celle-ci devra minimiser le plus possible les développements personnalisés afin de réduire les coûts. Si des modifications doivent être faites nous proposerons aussi un moyen d'estimer la taille de ces modifications à apporter à Odoo au moyen de Cosmic.

Notre méthode reposera d'abord sur une modélisation de Odoo au moyen d'un Feature model et de diagrammes BPMN. Ensuite nous détaillerons une méthodologie pour mettre en œuvre l'ERP Odoo dans une PME, cette méthodologie se concentrera sur les phases suivant l'analyse du besoin jusqu'à la contractualisation Nous aborderons aussi en détail la phase de prototypage et d'estimation de la taille de modification de Odoo.

Mots-clés : Odoo, ERP, point de fonction, Cosmic, BPMN, méthodologie ERP.

Abstract

Historically only large companies had the means to embark on expensive projects such as ERP projects. Moreover, ERP projects are risky because if they are poorly managed they can be quite expensive. In the context of SMEs, they can't afford to invest in too large projects and they can't afford the costs of failure of an ERP implementation

So, this master thesis addresses the complexity of the implementation of Odoo ERP in SMEs. We will also discuss the topic of variability with Feature models, and the topic of measuring the software changes size with Cosmic

Our goal is to provide an implementation methodology for the Odoo ERP, that is simple and adapted for SMEs. The methodology will minimize custom developments to reduce costs. If changes are made to the ERP, we will also propose a way of estimating the size of these changes with Cosmic.

Our method will rely first on modeling Odoo using a Feature model and BPMN diagrams. Then we will detail a methodology to implement the Odoo ERP in an SME, this methodology will focus on the phases from the analysis until quotations. We will also discuss in detail the phase of prototyping and estimation of the Odoo modification size.

Keywords: Odoo, ERP, function points, Cosmic, BPMN, ERP methodology.

Remerciements

Je remercie Michaël Petit et Benoît Vanderose pour leur soutien et conseils précieux tout au long de ce mémoire.

Je tiens ensuite à remercier Philippe Lepot de la société Imara pour m'avoir offert l'opportunité de gagner en expériences professionnel et connaissances acquis ces deux années.

Finalement, je tiens à remercier ma famille et plus particulièrement ma mère sans qui je n'aurais pas fait tous ce chemin jusqu'à présent et ma grand-mère qui m'a accompagné pendant tous mes blocus.

Table des matières

Partie I Introduction 7
Chapitre 1: Introduction 8
1.1. Contexte et problématique 8
1.2. Objectif 8
1.3. Méthode 8
1.4. Structure 9
Partie II État de l'art 11
Chapitre 2: Les ERPs 12
2.1. L'ERP ODOO 13
Chapitre 3: Méthodologies de mise en œuvre des ERPs 19
3.1. A model-based methodology for early stages of ERP implementations in SMEs [Mordant, 2007] 19
3.2. Définition d'une méthodologie de mise en œuvre et de prototypage d'un progiciel de gestion d'entreprise (ERP) [Boutin, 2001] 20
3.3. Contribution à une méthodologie et une modélisation pour accompagner les petites entreprises dans l'étude de leur organisation afin de spécifier leurs besoins et sélectionner une solution ERP [Lacombe, 2015] 22
Chapitre 4: Les softwares product line et Feature Models 24
4.1. Software Product Line Engineering 24
4.2. Les features model 25
4.3. Liens avec les ERPs 25
Chapitre 5: La modélisation de processus au moyen de BPMN 26
5.1. Les sous-processus 27
5.2. Les « data objects » 27
5.3. Les différents types de « Gateways » [lucidchart, BPMN Gateway Types] 28
Chapitre 6: Estimation de projets IT avec COSMIC 31
6.1. La phase de la stratégie de mesurage 31
6.2. La phase d'arrimage 33
6.3. La phase de mesurage 35
6.4. Modèle générique d'un logiciel selon Cosmic 36
6.5. Travaux sur BPMN et Cosmic 37
6.6. Travaux sur Cosmic et les erp 38

Partie III Contribution 41
Chapitre 7: Exemple d'une modification de Odoo : ajouter un champ 42
7.1. Ajout d'un champ dans la fiche du client 42
7.2. Application de la réduction automatiquement 43
Chapitre 8: Critique des méthodologies existantes, et proposition d'une nouvelle méthodologie 44
8.1. Critique de [Mordant, 2007] 44
8.2. Critique de [Boutin, 2001] 44
8.3. Critique de [Lacombe, 2015] 45
8.4. Proposition d'une nouvelle méthodologie 46
Chapitre 9: Proposition d'une modélisation de Odoo 49
9.1. Feature model de Odoo 49
9.2. Sélection d'une configuration 49
9.3. Représentation BPMN de Odoo 50
9.4. Description des « data objects » 51
Chapitre 10: Prototypage 52
10.1. Spécification des modifications 52
10.2. Estimation des développements à effectuer 53
Partie IV Application 57
Chapitre 11: Premier cas – Customer discount 59
11.1. Analyse du besoin 59
11.2. Sélection d'une configuration 59
11.3. Test « Est-ce que tous les besoins sont couverts » ? 59
11.4. Prototypage 59
11.5. Contractualisation 60
Chapitre 12: Deuxième cas – Commandes récurrentes 61
12.1. Analyse du besoin 61
12.2. Sélection d'une configuration 61
12.3. Test « Est-ce que tous les besoins sont couverts » ? 61
12.4. Prototypage 62
12.5. Contractualisation 63
Chapitre 13: Troisième cas - Product Pack 64
13.1. Domain engineering 64
13.2. Application engineering 64
Partie V Conclusion 65
Chapitre 14: Conclusion 66
14.1. Perspectives d'évolution 66

Partie VI Sources 67
Partie VII Annexes 70
Annexe 1 Exemple BPMN 71
Annexe 2 Méthodologie 72
Annexe 3 Feature model de Odoo 73
Annexe 4 Modélisation BPMN de Odoo 74
Annexe 5 Description des data Object 81
Annexe 6 Modification pour l'exemple n°1 84
Annexe 7 Modification pour l'exemple n°2 86
Annexe 8 Exemple n°3 89

Table des illustrations

Figure 1 Architecture d'un ERP selon [Lacombe, 2015] 12
Figure 2 Aperçus de l'interface de Odoo 15
Figure 3 Exemple d'un workflow [Odoo, Workflow] 17
Figure 4 Méthode de Boutin 20
Figure 5 Phase de prototypage [Boutin, 2001] 21
Figure 6 Méthodologie de [Lacombe, 2015] 22
Figure 7 Représentation du SPLE d'après [Anquetil, et al., 2008] 24
Figure 8 Exemple d'un Feature model 25
Figure 9 Exemple BPMN 26
Figure 10 Exemple sous-processus BPMN 27
Figure 11 Exemple Data Object 27
Figure 12 Différents type de Gateways source [Object Management Group, 2013] 28
Figure 13 Exemple Gateway Exclusive 28
Figure 14 Exemple Gateway Inclusive [Object Management Group, 2013] 29
Figure 15 Exemple Gateway EventBased [Object Management Group, 2010] 29
Figure 16 Exemple Gateway Prallel 30
Figure 17 Aperçu de la méthode Cosmic [Cosmic, 2016] 31
Figure 18 Utilisateur fonctionnel et frontière [Cosmic, 2016] 32
Figure 19 Grille de mesurage [Cosmic, 2015] 35
Figure 20 Exemple de EPC [Téllez, 2009] 38
Figure 21 Exemple de eEPC [Téllez, 2009] 38
Figure 22 Exemple des entrées [Téllez, 2009] 39
Figure 23 Méthodologie de [Lacombe, 2015] 45
Figure 24 Notre méthodologie 46
Figure 25 Feature model de Odoo 49
Figure 26 Exemple ajout et suppression 50
Figure 27 Exemple modification 51
Figure 28 Exemple de description d'un « data object » 51
Figure 29 Modifications du chapitre 7 52
Figure 30 Exemple de mouvements de données 54
Figure 31 Notre méthodologie 58
Figure 32 Configuration pour l'exemple 1 59
Figure 33 Configuration pour l'exemple 2 61
Figure 34 Configuration exemple 3 64

Partie I
INTRODUCTION

Chapitre 1: Introduction

1.1. Contexte et problématique

Ce mémoire aborde la complexité de la mise en œuvre d'un ERP dans des PME. En effet, historiquement seules les grandes entreprises avaient les moyens de se lancer dans des projets couteux que sont les projets ERP. De plus, les projets ERP sont risqués car s'ils sont mal menés ils peuvent coûter assez cher. Dans le contexte des PME, celles-ci n'ont pas les moyens d'investir dans de gros projets et ne peuvent pas se permettre d'assumer les coûts de l'échec d'une implémentation d'un ERP

En effet, bien qu'un ERP soit personnalisable et adaptable, il persiste un problème récurrent dans l'implémentation d'un ERP. Ce problème est de faire un compromis entre les exigences d'une entreprise et les processus que l'ERP propose.

De plus, dans le contexte des PME, celles-ci n'ont pas les moyens d'investir beaucoup d'argent dans de l'analyse, celle-ci doit donc être assez concise.

Dans ce mémoire nous aborderons aussi le thème de la variabilité, et de la mesure de la taille des modifications. Nous nous intéresserons à la taille des modifications car généralement la taille d'installation et de configuration dépend généralement de la taille de l'entreprise et peut être plus facilement estimée avec de l'expérience. Par contre estimer la taille d'une modification ne peut se faire seulement qu'en connaissant la taille de l'entreprise, celle-ci dépend de l'écart entre les processus de l'entreprise et les processus de l'ERP. Mesurer la taille de modification d'un ERP pour convenir à une entreprise permet donc de savoir si l'implémentation de l'ERP sera compatible avec l'entreprise ou si cette implémentation sera trop coûteuse.

Pour ce mémoire nous avons choisi d'utiliser l'ERP Odoo.

1.2. Objectif

Notre objectif est donc d'apporter une méthodologie d'implémentation pour l'ERP Odoo simple et adaptée pour les PME. Celle-ci devra minimiser au maximum les développements personnalisés afin de réduire les coûts. Si des modifications doivent être faites nous proposerons aussi un moyen d'estimer la taille de ces modifications à apporter à Odoo afin de se rapprocher au plus près du processus de l'entreprise.

Quantifier la taille de l'installation, l'importation des données et la paramétrisation de l'ERP ne fait pas partie du périmètre de ce mémoire.

1.3. Méthode

Notre méthode reposera d'abord sur une modélisation de Odoo au moyen d'un Feature model et de diagrammes BPMN. Ensuite nous détaillerons une méthodologie pour mettre en œuvre l'ERP Odoo dans une PME, cette méthodologie se concentrera sur les phases suivant l'analyse du besoin jusqu'à la contractualisation. La phase d'implémentation et de paramétrisation venant après la contractualisation n'est pas couverte par ce mémoire. Nous aborderons aussi en détail la phase de prototypage et d'estimation de la taille de la modification de Odoo.

1.4. Structure

Ce mémoire contient cinq parties : cette partie est dédiée à l'introduction, la partie suivante sera consacrée à notre état de l'art, puis viendra notre partie contribution, ensuite viendra une partie contenant nos cas pratiques et finalement une conclusion.

Tout d'abord, notre partie état de l'art détaillera les différents outils utilisés pour ce mémoire. Nous commencerons par une brève explication de ce qu'est un ERP, puis nous détaillerons la composition de l'ERP Odoo. Ensuite, nous ferons le tour de plusieurs méthodologies d'implémentation, puis nous introduirons brièvement ce qu'est le « software product line engineering » et le lien avec un ERP. Après cela, nous détaillerons ce qu'est BPMN. Et pour finir nous présenterons la méthode de mesure de logiciel par points de fonction Cosmic et son application dans deux travaux.

Ensuite dans notre partie contribution, nous présenterons un exemple de modification de Odoo, puis nous critiquerons les méthodologies vues dans notre état de l'art afin de proposer la nôtre. Nous proposerons finalement une méthode pour évaluer la taille d'une modification de Odoo.

Pour finir, nous appliquerons notre méthodologie dans trois cas pratiques et nous terminerons par une conclusion.

Partie II
ÉTAT DE L'ART

Chapitre 2: Les ERPs

Les ERPs (Enterprise Resource Planning, PGI pour Progiciel de Gestion Intégrée en français) tiennent leurs origines dans les MRP [Téllez, 2009], les MRP étaient des logiciels de gestion de chaines de production et nécessitaient déjà le regroupement de plusieurs domaines de compétences pour fonctionner (stock, provisions, commandes, …) Au fil du temps des fonctionnalités se sont ajoutées pour se généraliser dans d'autres domaines. Le principe d'un ERP est d'avoir un logiciel regroupant toutes ou plusieurs fonctionnalités de l'entreprise et fonctionnant sur une base de données unique.

Selon [Boutin, 2001] un ERP est un système d'information qui repose sur une seule base de données ce qui permet une unicité de l'information, cette base de données unique permet aussi une mise à jour en temps réel de l'information. En effet étant donné que l'on a qu'une base de données, lorsqu'on modifie une donnée dans une partie de l'ERP cette modification est disponible dans toutes les autres parties de l'ERP.

De plus une caractéristique de l'ERP est qu'il contient tout ou une bonne partie du business de l'entreprise et qu'il est conçu par un concepteur unique. L'utilisation du logiciel permet aussi d'avoir une vue d'ensemble sur les données.

Selon [Lacombe, 2015], un ERP permet de régler le problème de l'hétérogénéité des logiciels des entreprises en regroupant ces logiciels en une seule solution modulaire. Cette solution modulaire permet aux entreprises de choisir de quelles fonctionnalités leur entreprise a besoin. De plus ces modules sont tous articulés autour d'une seule base de données ce qui permet d'avoir les avantages cités dans la définition de Boutin. Lacombe résume un ERP par l'illustration suivante :

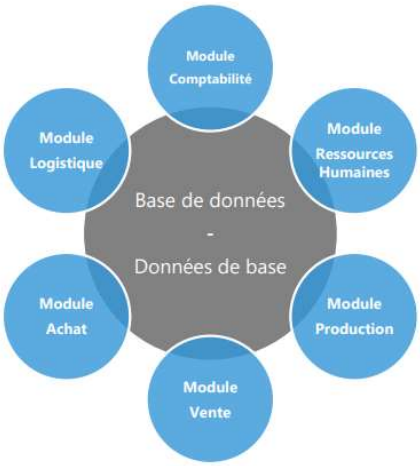


Figure 1 Architecture d'un ERP selon [Lacombe, 2015]

2.1. L'ERP ODOO[1]

L'ERP Odoo est issu d'une entreprise Belge, et dans ses premières versions en 2005 celui s'appelait Tiny ERP puis a ensuite été renommé OpenERP pour enfin s'appeler Odoo aujourd'hui.

Odoo existe en deux versions, une version Enterprise hébergée sur les serveurs de Odoo qui est uniquement éditable par eux, et une version Community que l'on peut installer gratuitement sur un serveur personnel et ainsi le modifier à sa guise. Odoo contient entre autres les fonctionnalités suivantes :

Gestion des achats : permet de prendre en charge les achats de fournitures d'une entreprise
Gestion des ventes : permet de prendre en charge les ventes de biens ou services de l'entreprise
Gestion de projets : propose une gestion de projet standard
MRP : permet de gérer des lignes de production, gestion des commandes, planification des commandes
Gestion de stock : permet une gestion des fournitures d'une entreprise, peux gérer des entrepôts multiples et les mouvements entre ces entrepôts.
Points de vente : propose une interface spéciale pour simuler une caisse enregistreuse pour faire de la vente au comptoir
Ressources Humaines : permet de gérer ce qui concerne les employés.
Comptabilité et Finance : permet de gérer les mouvements financiers, faire des rapports comptables, etc.
CRM : permet des faire de la gestion de clientèle, des prospections, des historiques d'appels, des campagnes marketings

Facturation : permet la gestion de la facturation des clients et fournisseurs.

2.1.1. Odoo Community

Dans ce mémoire nous travaillons donc uniquement avec la version Community. Pour pouvoir utiliser la version Community nous devons donc installer Odoo sur notre propre infrastructure (ou celle du client).

Cette installation consiste en l'installation du SGBD[2] PostgreSQL[3] et d'une application Python. Cette application est articulée autour d'une architecture MVC (pour Model-View-Controller). L'architecture MVC permet de séparer son application en trois parties, le modèle, la vue et le contrôleur. Le modèle représente la partie qui gère l'accès aux données, la vue permet de présenter ces données à l'utilisateur, et enfin le contrôleur contient la logique de l'application et fait l'intermédiaire entre la vue et le modèle.

Cette application consiste en un serveur Python (appelé backend[4] par la suite), celui-ci contient le « contrôleur », et un ORM[5] permettant de couvrir la partie « modèle ». Ce serveur

[1] Odoo: https://www.odoo.com/
[2] SGBD : Système de gestion de base de données
[3] PostgreSQL : https://www.postgresql.org/
[4] Backend : Ce sont tous les éléments que l'utilisateur ne voit pas mais qui font fonctionner l'applications

fournit une interface web écrite en Javascript qui couvre la partie « vue » (appelée frontend[6] par la suite).

Pour pouvoir modifier Odoo nous devons créer des modules. Ces modules sont composés de fichiers python et XML. Les fichiers Python permettent de modifier le code métier du backend. Les fichier XML sont, pendant l'installation d'un module, chargés en base de données afin de modifier l'interface frontend.

Nous allons maintenant présenter tout d'abord le frontend JavaScript et les différents éléments qui le composent, et ensuite nous présenterons le backend et ce que l'on peut modifier dans celui-ci.

---

[5] ORM : Object-relational mapping, service qui permet d'accéder à une base de données relationnelle uniquement en manipulant des objets.

[6] Frontend : Ce sont les éléments que l'on voit à l'écran, les éléments avec lesquelles l'utilisateur interagit

2.1.2. Frontend JavaScript

Le frontend JavaScript est l'interface présentée à l'utilisateur, elle représente la partie « vue » de l'architecture MVC.

Tout le comportement de cette interface se retrouvant dans la base de données, l'interface peut être alors modifiée sans toucher au code de Odoo, cependant nous pouvons créer des fichiers XML qui à l'installation du module seront chargés dans la base de données. L'avantage des fichiers XML est de pouvoir être intégrés dans un module pour ensuite répliquer cette modification sur plusieurs installations.

La modification du frontend peut être effectuée au moyen de plusieurs éléments. Avant de les présenter voici un aperçu du frontend avec les éléments qui le compose :

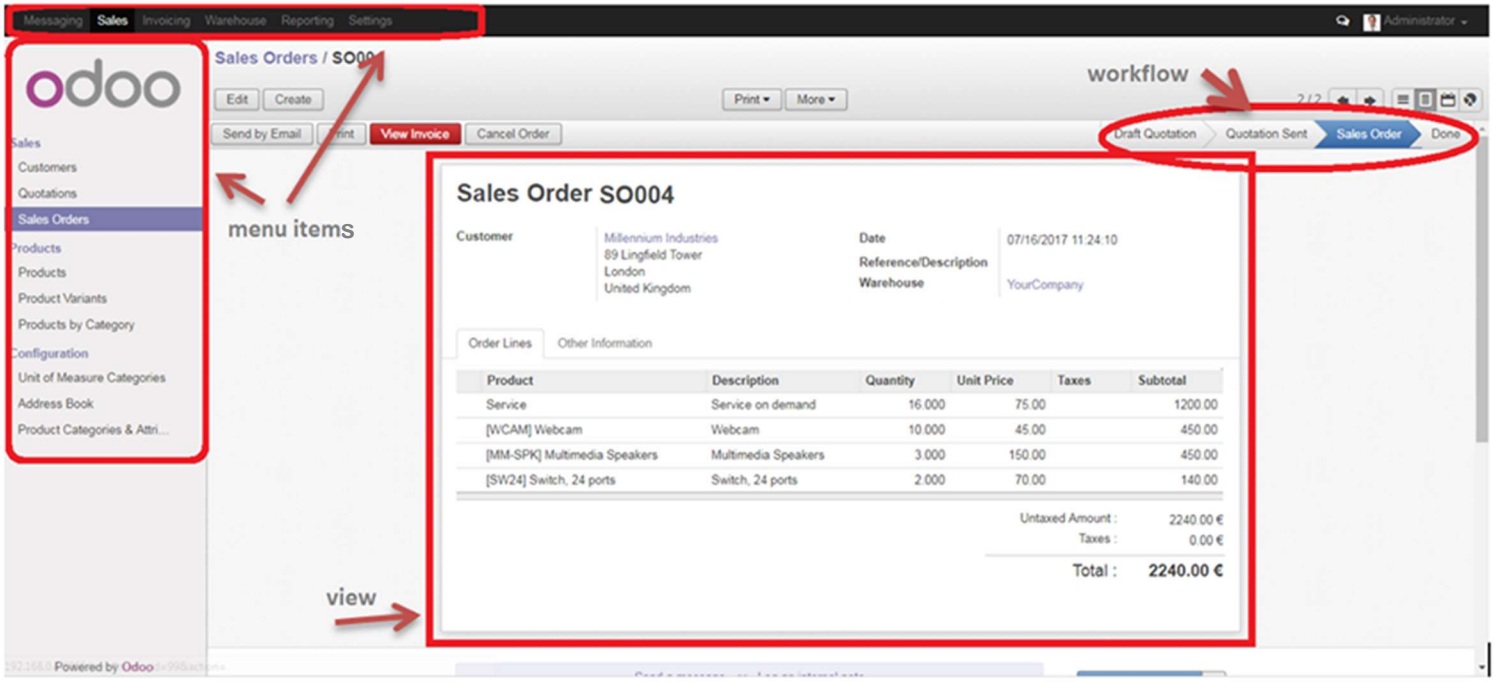


Figure 2 Aperçus de l'interface de Odoo

Sur cette capture nous pouvons voir des « menus items » en haut à gauche, au centre nous avons la « view » et au-dessus de la « view » un « workflow ».

Nous allons maintenant détailler ces différents composants dans les points suivants.

2.1.2.1. Les views

Selon [Odoo, Views] les « views » sont les éléments qui présentent les données à l'utilisateur dans l'interface Web, une « view » est toujours liée à un objet que cette « view » représentera, et il en existe 6 types :

1. Les « list views » : cette « view » permet de lister tous les enregistrements d'un objet, par exemple lister toutes les commandes. Elles peuvent permettre aussi de pouvoir éditer les enregistrements directement dans la liste.
2. Les « form views » : cette « view » permet d'afficher un formulaire permettant de modifier les valeurs des champs d'un enregistrement.
3. Les « graph view » : c'est une « view » qui permet de visualiser un champ d'une collection d'objets sous la forme d'un graphique.
4. Les « kaban view » : c'est une alternative à la « list view », celle-ci permet de présenter les différents enregistrements sous forme de cadres. Cette vue permet aussi de classer ces cadres dans différents états d'un workflow. Par ex : To Do, Doing, Done

Il existe aussi des « calendar view » et des « gantt view » mais celles-ci sont plus spécifiques.

Toutes les « views » ont un aspect et un comportement hérité d'une « view » générique, les modifications des « views » se font donc assez facilement car il ne faut écrire que ce que l'on veut modifier de la vue générique.

Les vues sont spécifiées dans un langage de Template appelé QWeb [Odoo, Qweb]. Celui-ci est proche du langage Html mais il permet plus de possibilités telles que faire des tests au moyen de « if », des boucles et inclure directement des données de la base de données.

Etant donné que les pages sont rédigées au moyen du langage XML, on peut facilement modifier leur code grâce à Xpath[7]. Xpath permet de faire des recherches dans un fichier XML, puis ensuite de pouvoir soit ajouter ou remplacer une portion de code XML. Un exemple d'utilisation de Xpath est montré dans le chapitre 7:.

2.1.2.2. Les menu Items

Les « menu items » sont les boutons qui sont soit dans la barre du dessus, soit sur la barre latérale. Chaque menu item permet d'appeler une action.

2.1.2.3. Les actions [Odoo, Actions]

Les actions permettent de déclencher des événements du système. Il en existe 5 types :

1. Windows action : cette action permet d'afficher une « view »
2. Url action : permet de rediriger vers une url externe
3. Server action : permet de lancer l'exécution d'un code sur le serveur Python
4. Report action : permet de déclencher la génération d'un rapport, les rapports seront vus au point 2.1.2.6.
5. Client action : permet d'appeler une fonction dans le frontend JavaScript

[7] https://fr.wikipedia.org/wiki/XPath

2.1.2.4. Les Crons

Les Crons servent à appeler une action à un intervalle déterminé, comme les Crons Unix

2.1.2.5. Les Workflows [Odoo, Workflow]

Un workflow est un outil qui nous permet d'implémenter un processus sous forme de machine à état. Un workflow est toujours associé à un objet qui contient un champ « state », ce champ contiendra l'état courant de l'objet. Pour créer un workflow, il faut créer les différents états, dont un état de début et un ou plusieurs états de fin. Ensuite il faut déclarer toutes les transitions possibles pour passer d'un état à l'autre. A chaque transition on peut lier une action qui sera appelée lors de la transition, et par conséquent cette action permettra d'afficher une vue ou exécuter du code sur le serveur.

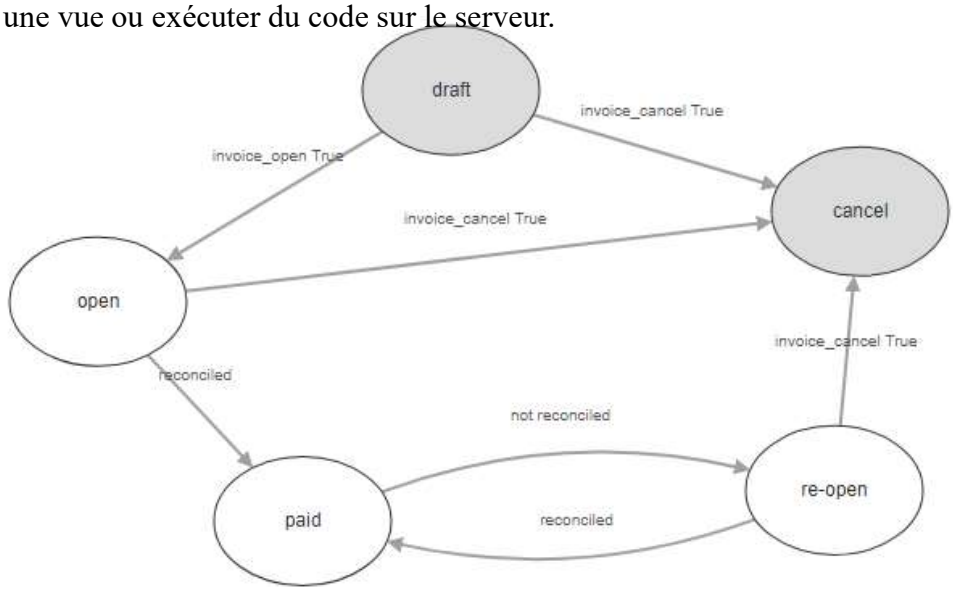


Figure 3 Exemple d'un workflow [Odoo, Workflow]

Les workflows sont assez couteux à implémenter, en général on ne fait que des modifications telles qu'ajouter ou supprimer des états ou des transitions.

2.1.2.6. La génération des rapport PDF

Il est aussi possible de générer des rapports. Un rapport est un fichier PDF que l'on peut implémenter grâce au langage QWeb. En fait, Odoo utilise le même moteur de Template que le frontend JavaScript afin de générer des pages Htmls à partir de vues QWeb et ensuite utilise l'utilitaire wkhtmltopdf[8]. Wkhtmltopdf est un petit utilitaire qui permet de convertir une page Html en PDF.

[8] https://wkhtmltopdf.org/

2.1.3. Backend Python

Comme vu au point 2.1.1 le backend Python est séparé en deux parties : le contrôleur et le modèle.

2.1.3.1. Le contrôleur

Le contrôleur est l'élément central de Odoo, il permet d'interagir avec l'ORM et il fournit l'interface web du frontend. Vu qu'il joue le rôle de contrôleur il réagit aux actions que l'utilisateur effectue dans l'interface frontend et fournit une réponse à ces actions en exécutant le code métier. Ce code métier peut accéder à l'ORM.

2.1.3.2. Le modèle

La partie modèle consiste en un ORM implémenté par Odoo. Un ORM permet d'interfacer une base de données relationnelle par des objets. Tout ce que le concepteur doit faire c'est de créer des objets et de relier les attributs de ces objets avec les attributs des tables de la base de données. L'ORM de Odoo se charge ensuite de récupérer, créer, modifier ou supprimer les données en base de données et de nous retourner ces données sous la forme des objets que l'on a créé. Le SGBD utilisé par Odoo est PostgreSQL.

2.1.4. Fonctionnement de base de Odoo

Au démarrage de Odoo le backend Python charge le serveur web, se connecte au gestionnaire de base de données, charge les fonctionnalités de base de Odoo et détecte les modules qui se trouvent dans le dossier « addons ».

L'interface web contient un utilitaire qui permet ensuite de créer, sauvegarder, restaurer et supprimer les bases de données, ce qui permet d'avoir plusieurs instances de Odoo sur un même serveur.

Une fois une base de données créée, l'interface permet de se connecter à son instance Odoo.

Une fois connecté sur son instance on peut parcourir les différents modules présents au démarrage du serveur afin de pouvoir installer ceux dont on a besoin.

2.1.5. Structure d'un module [Odoo,How To Backend]

Maintenant que nous avons abordé l'architecture de Odoo nous pouvons voir comment modifier celui-ci au moyen de modules :

Un module est composé de plusieurs fichiers Python et XML. La structure de base des fichiers d'un module Odoo est :

- Le fichier « __init__.py », qui contient les imports des fichiers Python du module
- Le fichier « __ openerp __.py », fichier qui décrit le module,
- Des fichiers Python et XML

Le fichier « __ openerp __.py » permet de déclarer le nom, la description, les dépendances et la liste des fichiers XML du module.

Les fichiers XML déclarent les données à importer dans la base de données à l'installation du module. Comme vu au point 2.1.2 modifier la base de données au moyen de fichiers XML permet aussi de modifier tous les éléments de l'interface frontend.

Les fichiers Python permettent de déclarer de nouveaux objets, leurs attributs et les méthodes qui contiennent le code métier de l'application.

Chapitre 3: Méthodologies de mise en œuvre des ERPs

Dans cette partie nous allons présenter trois publications qui parlent de méthodologie des ERPs. La première écrite par [Mordant, 2007] propose une première méthodologie d'implémentation d'ERP. Ensuite, nous regarderons du côté de la thèse de [Boutin, 2001], qui propose une autre méthodologie, et pour finir nous aborderons la thèse de [Lacombe, 2015] qui reprend notamment la thèse de Boutin afin de l'adapter aux PME.

3.1. A model-based methodology for early stages of ERP implementations in SMEs [Mordant, 2007]

Dans son mémoire, [Mordant, 2007] nous propose une méthodologie d'implémentation en trois parties : une première partie de définition du « Business Case » ou il analyse ce qui existe dans l'entreprise, puis une partie d'implémentation où l'ERP est peu à peu intégré à l'entreprise, et finalement une phase appelée « Operation » où l'ERP a remplacé l'ancien système.

3.1.1. Définition du « business Case »

Dans cette phase [Mordant, 2007] va tout d'abord faire une phase de préparation où il va collecter les exigences de l'entreprise, analyser ces exigences, les délais de retour sur investissement, le but et les impacts de l'adoption de l'ERP. Cette phase permettra l'élaboration d'un contrat.

Ensuite il se lance dans une phase intitulée identification et modélisation, où une fois le contrat signé il va définir le planning du projet et les équipes de projet, raffiner le but et les exigences et définir des KPI (indicateurs clés de performance). Enfin, il finit par modéliser les principaux processus métier de l'entreprise et acquiert une vision globale de l'entreprise.

3.1.2. Implémentation

Cette partie est divisée en trois phases, une phase d'alignement où l'ERP va être installé sur un système de test, puis configuré selon le « Business Case » et où les processus de l'entreprise vont être modifiés pour convenir au nouvel ERP.

Il en suit une phase d'installation, où l'ERP va être installé dans l'entreprise cliente, suivi par des tests et des corrections d'erreurs. Il va ensuite former les utilisateurs à l'utilisation du nouveau système, et suite à cette formation il peut encore raffiner les personnalisations du système pour mieux convenir aux utilisateurs.

Une fois la phase d'installation terminée, il refait une session de formation avec les utilisateurs finaux pour valider et finaliser la configuration de l'ERP. A la fin de cette phase l'ERP est alors en production.

3.1.3. Opération

Les tâches réalisées dans la partie « opération » sont toutes les tâches qui suivent la mise en production. Elle est constituée d'adaptations mineures qui pourraient encore survenir une fois que les utilisateurs sont habitués au système. En effet, dans cette partie des données sur le fonctionnement de l'ERP sont collectées et on peut éventuellement apporter des modifications pour améliorer son utilisation. Le contenu de cette phase dépend aussi de ce qui a été convenu dans le contrat.

3.2. Définition d'une méthodologie de mise en œuvre et de prototypage d'un progiciel de gestion d'entreprise (ERP) [Boutin, 2001]

Pour notre méthodologie nous nous sommes notamment intéressés à la méthodologie de Pascal Boutin. Il propose une méthodologie qui se résume au schéma suivant :

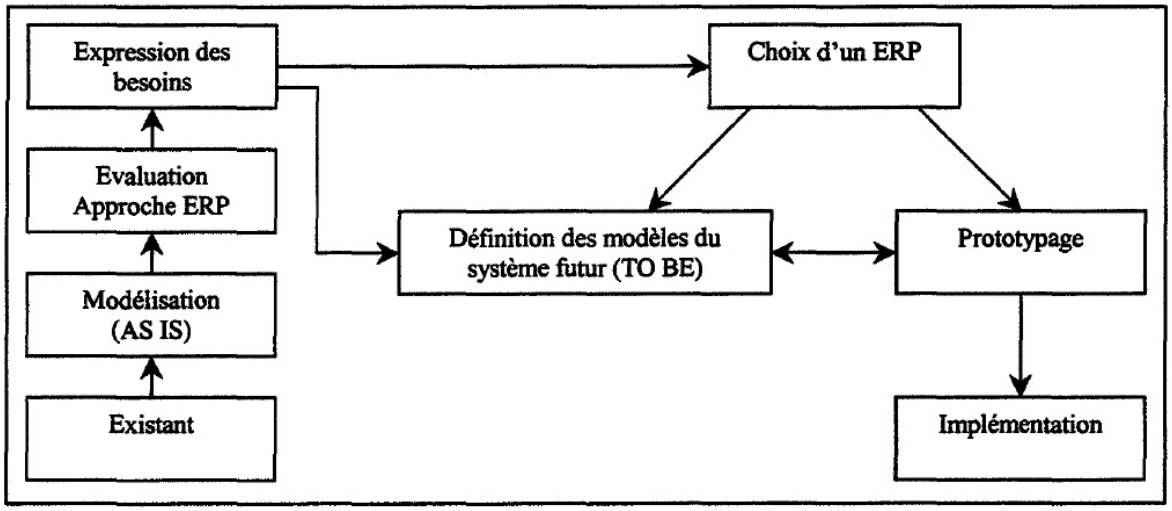


Figure 4 Méthode de Boutin

Il décompose ensuite sa méthodologie en six phases (la phase d'implémentation n'est pas couverte par sa thèse) :

3.2.1. La phase de modélisation

Dans cette phase il modélise les principaux aspects existants de l'entreprise de manière macroscopique, sans trop de détails, ceci afin de comprendre quelle est l'entreprise et ce qui la caractérise comparée aux autres entreprises. Cette phase permet aussi à l'entreprise d'exprimer ce qu'elle voudrait changer dans ses processus existants.

3.2.2. La phase d'évaluation

Dans cette phase il critique les modèles obtenus dans la phase précédente. Il détecte les données redondantes qui pourront être unifiées dans la base de données unique de l'ERP. Il fait aussi attention de savoir si tous les processus de l'entreprise sont compatibles à une approche ERP.

3.2.3. La phase d'expression des besoins

Dans cette phase il collecte les désirs de l'entreprise en termes d'évolutions organisationnelles, ce vers quoi elle veut se diriger. Cette étape représente la vision idéale du système futur. Cette phase permet de réaliser un cahier des charges.

3.2.4. La phase de choix de l'ERP

Celle-ci peut se dérouler en parallèle avec la phase suivante mais doit être commencée avant la phase suivante. Dans cette phase il choisit un système ERP qui conviendra le mieux à l'entreprise. Il effectue cela à partir d'un questionnaire, l'ERP choisi sera celui qui aura le plus de réponses positives à ce questionnaire.

3.2.5. La phase de définition des modèles du système futur

Ici il s'agit de modifier le système existant afin de convenir au modèle de l'ERP, cette phase permet aussi une réingénierie des processus de l'entreprise afin d'optimiser l'utilisation de

l'ERP. Dans cette phase il ne recherche pas la solution optimale mais celle qui fonctionnera le mieux avec l'ERP. Il a choisi de réaliser cette phase en même temps que la phase de prototypage.

3.2.6. La phase de prototypage

Enfin dans cette phase il spécifie ce qui doit être réalisé pour implémenter l'ERP, c'est à dire, à partir du modèle TO-BE de l'entreprise et du modèle de l'ERP il génère le modèle final de l'ERP, un guide de paramétrage et un compte rendu. Ce compte rendu contient toutes les décisions et les choix qui ont été faits pendant le prototypage.

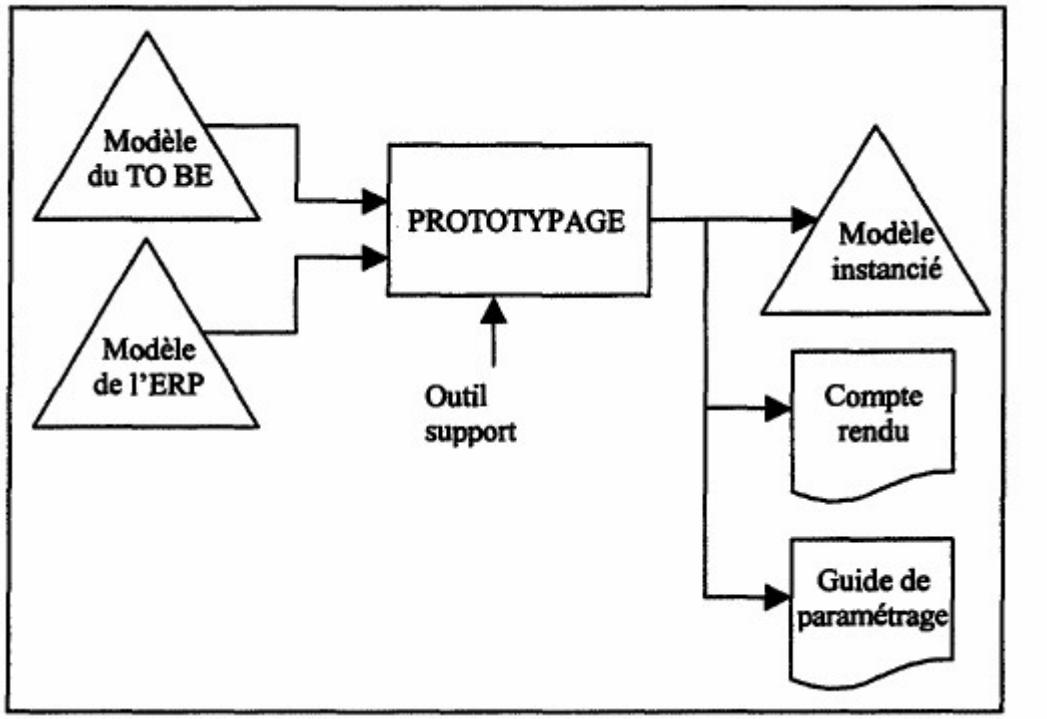


Figure 5 Phase de prototypage [Boutin, 2001]

3.3. Contribution à une méthodologie et une modélisation pour accompagner les petites entreprises dans l'étude de leur organisation afin de spécifier leurs besoins et sélectionner une solution ERP [Lacombe, 2015]

La thèse de [Lacombe, 2015] s'inspire entre autres de la thèse de Boutin pour proposer sa méthodologie de mise en œuvre d'ERP, cette démarche a été réalisée afin de correspondre aux petites entreprises. Il sépare sa méthodologie en trois phases : la phase d'étude avant-projet, la phase d'étude de l'organisation et de sélection de l'ERP, et la phase d'implémentation. Sa thèse repose uniquement sur les deux premières phases.

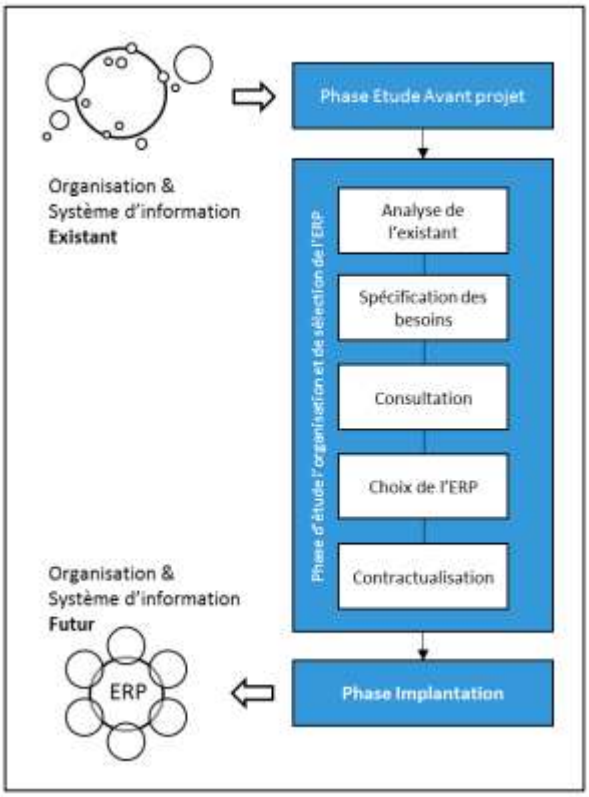


Figure 6 Méthodologie de [Lacombe, 2015]

3.3.1. La phase d'étude avant-projet

[Lacombe, 2015] sépare cette phase en 3 étapes, dans la première étape appelée « Veille du marché des ERP » consiste à présenter une vision de ce qu'est un ERP à l'entreprise. Elle comporte une présentation des enjeux et risques liés à la mise en place de l'ERP, une présentation des ERPs disponibles sur le marché et leur fonctionnalité ainsi que leur coût. Elle présente aussi la démarche utilisée. Cette étape permet aussi d'étudier la faisabilité du projet.

La deuxième partie « étude d'opportunités » s'intéresse à l'organisation du projet. Elle permet de définir les objectifs du projet, le périmètre couvert par le projet, et une première étude macroscopique des processus de l'entreprise. Il constitue ensuite une équipe et définit les contraintes et le planning à respecter.

Ces deux étapes permettent d'établir la dernière étape la « réunion de lancement », cette réunion permet de faire se rencontrer tous les acteurs du projet afin de présenter ce qui a été établi dans les deux étapes précédentes. Elle permet aussi de valider les rôles et acteurs du système, et les missions des acteurs externes.

3.3.2. La phase d'étude de l'organisation et de sélection de l'ERP

Cette phase est séparée en cinq parties : l'analyse de l'existant, la spécification des besoins, la consultation, le choix de l'ERP, la contractualisation

3.3.2.1. Analyse de l'existant

La première partie est l'analyse de l'existant : il fait cela en deux activités. Dans la première il analyse la structure de l'entreprise et son contexte, le marché ciblé par cette entreprise et l'offre de l'entreprise.

La deuxième activité est l'analyse et la modélisation de l'entreprise « as-is », cette modélisation comprend l'analyse des ressources humaines et fonctions de l'entreprise, la modélisation du système piloté, du système de pilotage, du système d'information, des interactions et des processus métiers.

3.3.2.2. La spécification des besoins

Dans cette partie il commence par modéliser le modèle « as-wished », ce modèle contient la modélisation du système piloté de l'entreprise au moyen d'actigrames, la modélisation du système de pilotage avec une grille GRAI, la modélisation du système d'information avec UML, et la modélisation des processus métiers avec un modèle d'interaction. Ensuite avec cette modélisation « as-wished », il définit les spécifications fonctionnelles de l'entreprise.

Cette partie aboutira à un cahier des charges, et une grille de réponses à soumettre au concepteur d'ERP dans l'étape suivante. Cette grille de réponses est « *un référentiel permettant d'analyser, d'évaluer et de comparer sur les mêmes critères les offres des concepteurs* » [Lacombe, 2015] .

3.3.2.3. Consultation restreinte des éditeurs et Choix de l'ERP

Dans cette étape, il sélectionne les éditeurs d'ERP qui pourraient convenir pour le projet, il leur transmet le cahier des charges et la grille de réponses. Il choisit ensuite un ERP parmi les propositions des concepteurs.

3.3.2.4. Contractualisation

Une fois l'éditeur d'ERP identifié, il entame une partie de négociation commerciale en vue d'aboutir à une contractualisation. La négociation permet de déterminer quelles prestations et engagements seront fournis entre les deux parties.

Chapitre 4: Les softwares product line et Feature Models

4.1. Software Product Line Engineering

L'apparition du Software Product Line Engineering (SPLE) peut être comparé aux lignes de production de voitures. En effet les voitures sont construites sur un modèle générique auxquel on rajoute des pièces en fonction des options que l'on a commandées. [Ben Rhouma, 2012]

En effet, le SPLE consiste à avoir un logiciel de base qui peut varier au lieu d'avoir plusieurs logiciels semblables. Le SPLE consiste à modéliser les parties du logiciel qui sont communes à toutes les variantes et à déterminer ce qui caractérise les différentes variantes. [Beuche, et al., 2007]

Pour ce faire le SPLE est divisé en deux phases : le « domain engineering » qui consiste à rechercher les différentes parties de logiciel et le « application engineering » qui consiste à sélectionner une configuration du logiciel. De plus ces deux phases possèdent deux perspectives : dans la perspective du problème on se concentre sur la définition des besoins et la perspective de la solution se charge elle de répondre à ces besoins. [Anquetil, et al., 2008]

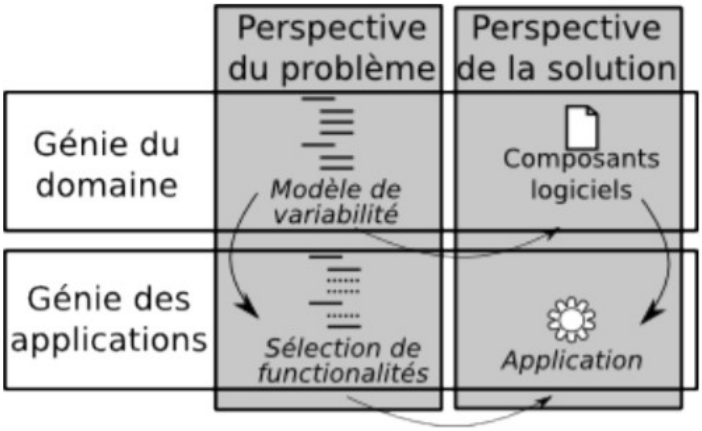


Figure 7 Représentation du SPLE d'après [Anquetil, et al., 2008]

4.1.1. Domain engineering

D'après [Anquetil, et al., 2008] *« l'analyse de domaine est le processus d'identification, collecte, organisation et représentation des informations pertinentes d'un domaine, en se basant sur l'analyse de systèmes existants* ».

Cette partie sert donc à rechercher les éléments communs au système et les différentes variantes possibles, la synthèse de ces variantes est schématisée dans un Feature model ; les Feature model seront vu au point 4.2. Chaque variante doit faire l'objet d'une composante logiciel.

4.1.2. Application engineering

Celle-ci consiste à sélectionner une configuration valide parmi les variantes trouvées dans la phase de « domain engenering » afin de correspondre au mieux au besoin.

Le processus de développement devient donc un processus de dérivation d'une nouvelle application à partir de la structure de base et en ajoutant les fonctions spécifiques nécessaires à l'application.

4.2. Les features model

Cette modélisation permet de représenter toutes les fonctionnalités du « domain engineering » en un seul diagramme. Ces fonctionnalités sont représentées sous forme d'arbres où chaque nœud représente une fonctionnalité. Il existe 4 types de nœuds :

- Les éléments obligatoires, c'est-à-dire si le feature parent est sélectionné ce feature doit être sélectionné aussi.
- Les éléments optionnels.
- Les alternatives, où une et une seule variante peut être choisie.

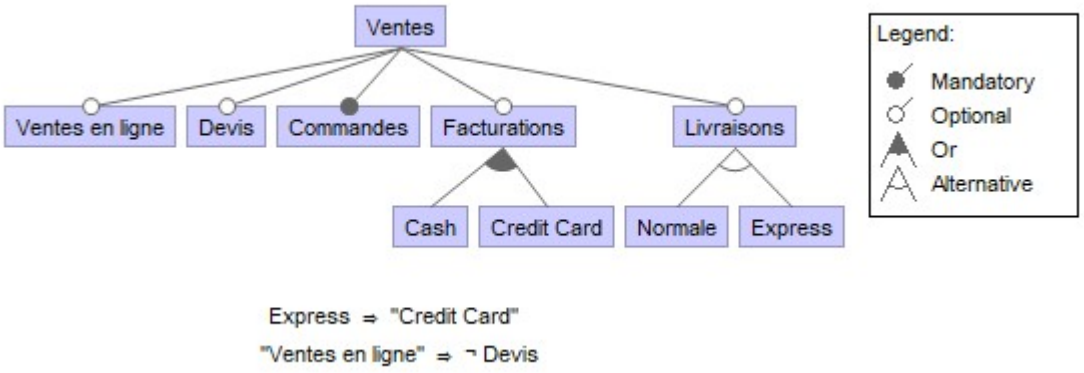


Figure 8 Exemple d'un Feature model

Les Or-feature, où au moins une variante doit être choisie. [Ben Rhouma, 2012]

Nous avons en plus deux autres contraintes :

- Les contraintes de dépendance, représentées par un signe implication (=>), cette contrainte implique que si la fonctionnalité avant l'opérateur est sélectionnée, celle après l'opérateur doit être sélectionnée aussi.
- Les contraintes d'incompatibilité, représentées par le signe de l'implication suivi de la négation (=> ¬), cette contrainte implique que si une des deux fonctionnalités est sélectionnée l'autre ne peut pas l'être non plus.

4.3. Liens avec les ERPs

Les ERPs peuvent être assimilés au Product line software, en effet, on a un composant de base auquel on peut ajouter des fonctionnalités et modifier le comportement de celui-ci au moyen d'installation de modules. Tous les modules peuvent donc être assimilés à une fonctionnalité du feature model. Ainsi choisir une configuration du feature model permet de connaitre quels modules doivent être installés dans l'ERP. De plus le feature model nous permet de nous assurer que la configuration choisie sera valide et donc quels modules installés seront compatibles entre eux.

Chapitre 5: La modélisation de processus au moyen de BPMN

BPMN est un standard de modélisation qui nous permet de représenter facilement les processus d'un logiciel ou d'une entreprise.

Tout comme dans les logigrammes (Flow chart en anglais), les activités sont représentées au moyen de rectangles aux coins arrondis et les contrôles de flux sont représentés par des losanges appelés « Gateway ». Une activité peut soit être une tâche ou un sous-processus. Les « Gateways » et « sous-processus » seront abordés plus en détail plus loin dans ce travail. Pour déterminer dans quel ordre les activités et les « Gateways » s'enchainent il faut les relier dans cet ordre avec des flèches continues appelée « sequence flow ». [Lucidchart], [Object Management Group, 2013]

BPMN permet aussi de représenter des événements, ceux-ci sont représentés par des ronds. Il existe trois types d'évènements ; les évènements de début représentés par un cercle avec un trait fin, les événements intermédiaires représentés par un cercle avec un trait double, et les événements de fin représentés avec un trait épais. Les événements BPMN les plus courants sont les événements de début. Dans nos diagrammes ceux-ci ont en plus une couleur, vert pour les évènements de début et rouge pour les évènements de fin, orange pour les événements intermédiaires. [Object Management Group, 2013]

Enfin, BPMN nous met en plus à disposition deux outils intéressants : les sous-processus et les « Data Objects », nous détaillerons ceux-ci avec l'exemple ci-dessous. Et nous finirons enfin par détailler les différents types de « Gateways ».

Voici un petit exemple de processus d'une commande dans une société d'e-commerce (Ce diagramme est disponible en annexe 1 pour plus de lisibilité)

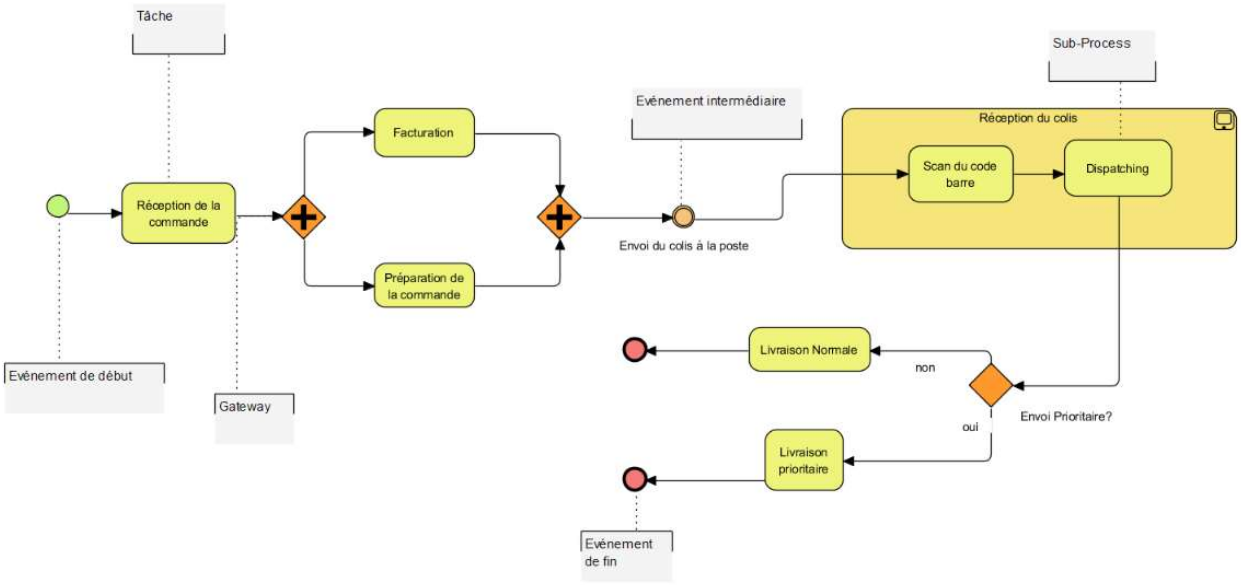
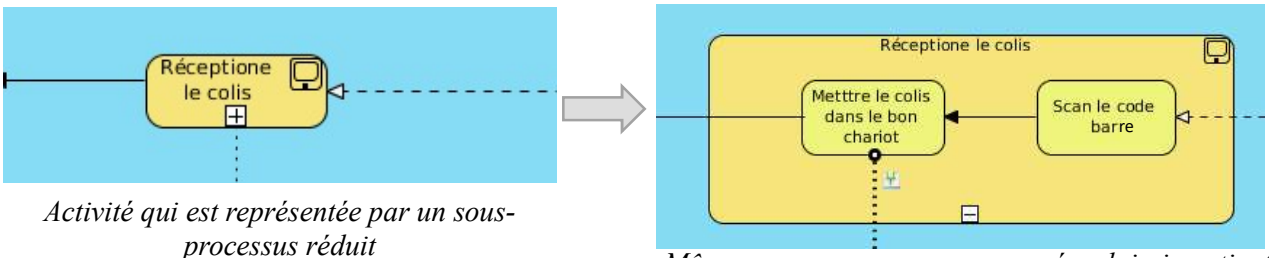


Figure 9 Exemple BPMN

5.1. Les sous-processus

Comme mentionné plus haut, les activités peuvent soit être des tâches ou des sous-processus. BPMN nous permet de représenter des sous-processus à l'intérieur d'une activité, d'un processus, ce qui nous permet d'avoir une hiérarchie dans nos processus. Cela nous permet de représenter un business avec tout d'abord des tâches simples (une représentation de haut niveau), et ensuite de détailler certaines de ces tâches avec des sous-processus plus détaillés.



Activité qui est représentée par un sous-processus réduit

Même sous processus non masqué, celui-ci contient des tâches simples

Figure 10 Exemple sous-processus BPMN

5.2. Les « data objects »

En plus de fournir des outils pour pouvoir représenter le déroulement des processus d'un logiciel, BPMN nous met à disposition un outil appelé « data object » afin de représenter les données manipulées par les processus.

Les « data objects » sont représentés par une feuille reliée avec une flèche pointillée aux activités qu'il utilise. Ces flèches sont appelées « data association ». On peut spécifier un état à un « data object », pour ce faire il suffit d'écrire l'état entre des crochets « [ ] ». Un « data object » peut être soit nécessaire à l'exécution d'une activité dans ce cas c'est un « Data Input », il peut être aussi produit par une fonction dans ce cas c'est un « Data Output ».

Un « data object » peut aussi être déplacé d'une activité à une autre. Ceci peut être fait soit en reliant le « data object » et un « sequence flow » par une ligne pointillée, soit en passant le « data objet » avec des « data association ».

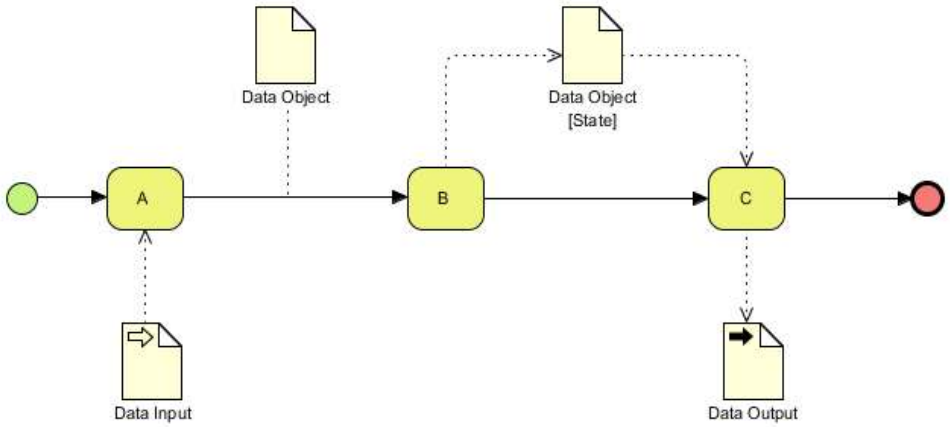


Figure 11 Exemple Data Object

5.3. Les différents types de « Gateways » [lucidchart, BPMN Gateway Types]

Dans BPMN les « Gateways » ne représentent pas que des points de choix (if), ils permettent non seulement de diriger le flux d'exécution d'une direction à une autre, mais ils permettent aussi de dupliquer (fork) et refusionner des flux. Une analogie du flux d'exécution peut être représenté par un jeton qui suit les flèches de flux et les « Gateways » peuvent être comparés à des aiguillages qui déterminent la direction de ce jeton.

Ci-dessous un aperçu des différents types de « Gateways » et leur représentation et ensuite leurs significations plus détaillées.

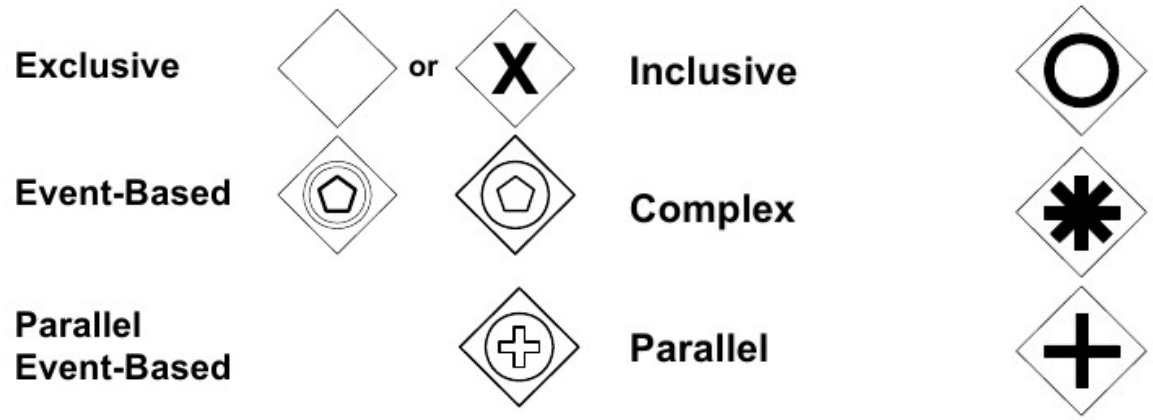


Figure 12 Différents type de Gateways source [Object Management Group, 2013]

5.3.1. Exclusive

Avec la « Gateway » exclusive on a une entrée et on ne peut sortir que par une seule sortie, cette sortie est définie en fonction du résultat de l'exécution d'une expression (une question/ un test). Elle est soit représentée par un losange vide ou peut être représentée par un losange contenant un « X » mais ce n'est pas obligatoire. L'exemple de la figure 13présente un choix ou soit on procède à une expédition normale soit à une expédition prioritaire, il est impossible de faire les deux. Un autre exemple est l'exemple 1 dans les pages précédentes.

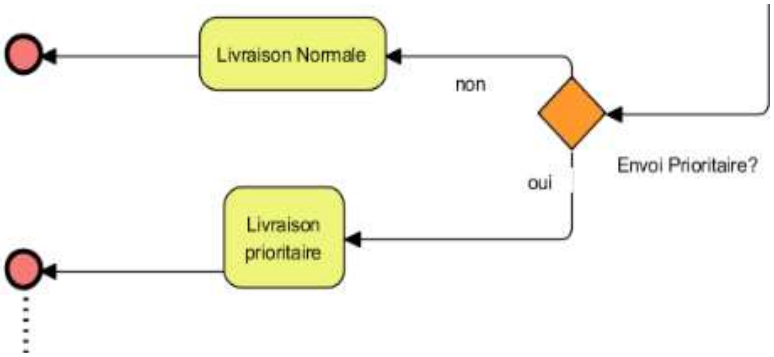


Figure 13 Exemple Gateway Exclusive

5.3.2. Inclusive

Cette « Gateway » est similaire à la « Gateway » exclusive sauf que celle-ci peut avoir plusieurs flux (duplications) de sortie. En effet, il existe des cas où plusieurs réponses pourraient résulter de l'évaluation de l'expression. Cependant, il faut s'assurer que le flux sortira par au moins une des sorties. Si besoin, une sortie par défaut peut être définie afin de

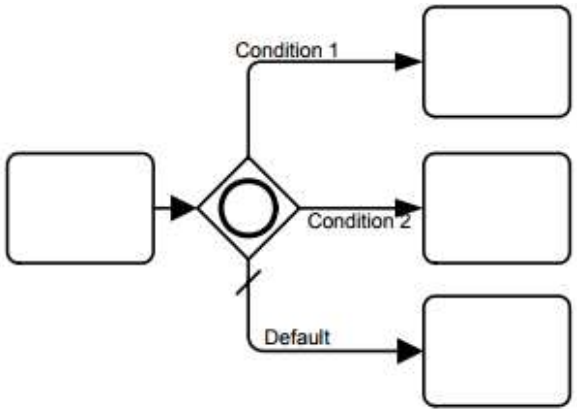


Figure 14 Exemple Gateway Inclusive [Object Management Group, 2013]

s'assurer qu'on aura toujours au moins une sortie pour le flux.

5.3.3. EventBased

Cette « Gateway » est aussi similaire à la « Gateway » exclusive sauf que la sortie ne dépend pas de l'évaluation d'une expression mais dépend d'un évènement qui peut survenir.

Dans l'exemple ci-dessous, une fois la commande d'une pizza effectuée on attend que soit l'évènement « Pizza reçue » se produise et dans ce cas on paye la pizza et on continue dans cet embranchement, soit une heure s'écoule et on redemande à la pizzeria si elle n'a pas oublié la commande, puis on retourne dans le cas avant la « Gateway ».

Cette « Gateway » est exclusive, cette à dire que pour un flux en entrée il n'y aura jamais qu'un seul flux en sortie, cette « Gateway » ne duplique jamais un flux, en effet, si plusieurs évènements correspondant aux possibilités de la « Gateway » surviennent, seul le premier sera pris en compte et tous les autres seront ignorés. Pour que la « Gateway » reprenne à nouveau en compte d'autres événements il faut qu'un nouveau flux arrive à son entrée. Dans l'exemple ci-dessous, si l'événement « 60 minutes écoulées » survient, mais que la pizza a déjà été reçue, on ne déclenchera pas la tâche « ask for the pizza », par contre, si on est dans le cas où une heure est passée, et que l'on a toujours pas reçu sa pizza, on passe par l'étape « ask for the pizza », ensuite, le flux reviendra au début de la « Gateway » et on pourra de nouveau recevoir l'événement "pizza reçue".

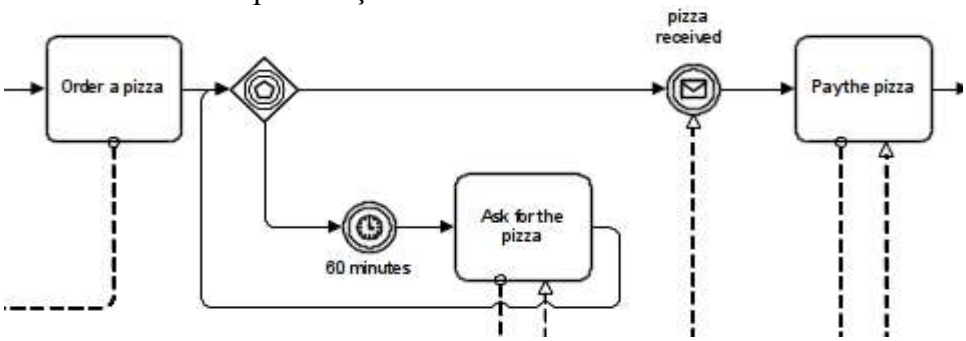


Figure 15 Exemple Gateway EventBased [Object Management Group, 2010]

5.3.4. Complex

Comme son nom l'indique cette « Gateway » permet de gérer des expressions plus complexes comme exiger qu'au moins deux des trois conditions possibles soit validées pour que ces deux branches reçoivent un flux. L'utilisation de cette « Gateway » est assez rare car généralement une combinaison des autres Gateways suffit pour représenter la majorité des cas.

5.3.5. Parallel

Cette « Gateway » se différencie des autres car son fonctionnement ne dépend pas d'un test ou d'un événement. Elle permet de créer des flux supplémentaires (fork) sur chacune de ses sorties, c'est à dire que pour chaque flux en entrée il y a des flux qui sortent par chacune de ses sorties. Cette « Gateway » peut être aussi utilisée pour refusionner des flux, dans ce cas la « Gateway » a plusieurs entrées et une seule sortie. Dans l'exemple ci-dessous, une fois la commande réceptionnée on lance à la fois la facturation et la préparation de la commande, et

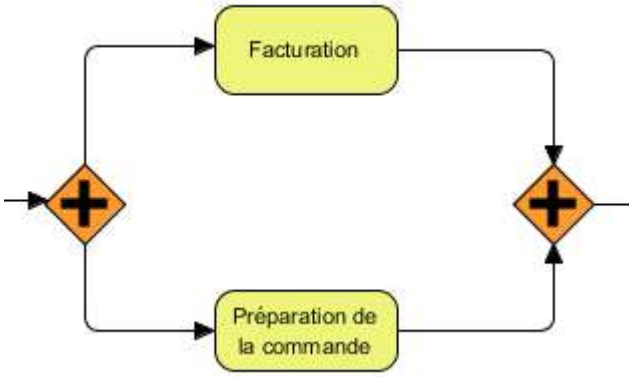


Figure 16 Exemple Gateway Prallel

on attend que les deux soient finies avant de continuer avec la livraison.

5.3.6. Parallel Event Based

Elle a le même comportement que la « Gateway Event based » sauf que celle-ci n'est pas exclusive elle va recréer de nouveaux flux si plusieurs événements surviennent.

Chapitre 6: Estimation de projets IT avec COSMIC

Cosmic est une méthode de mesure de logiciel par points de fonction. Cette méthode permet de mesurer plusieurs types de logiciels comme les applications business, le temps réel, les logiciels multicouche. [Cosmic, 2016]

Celle-ci consiste à mesurer des « fonctionnalités utilisateur Requises » (FUR) d'une application. Une « *FUR indiquent ce que le logiciel doit accomplir pour ses utilisateurs, en termes de tâches et services* » [Cosmic, 2016];

L'application de Cosmic se fait en 3 phases :

- La phase de la stratégie de mesurage
- La phase d'arrimage
- Et enfin la phase de mesure

Avant de les détailler voici un aperçu tiré du manuel de mesure de Cosmic [Cosmic, 2015]

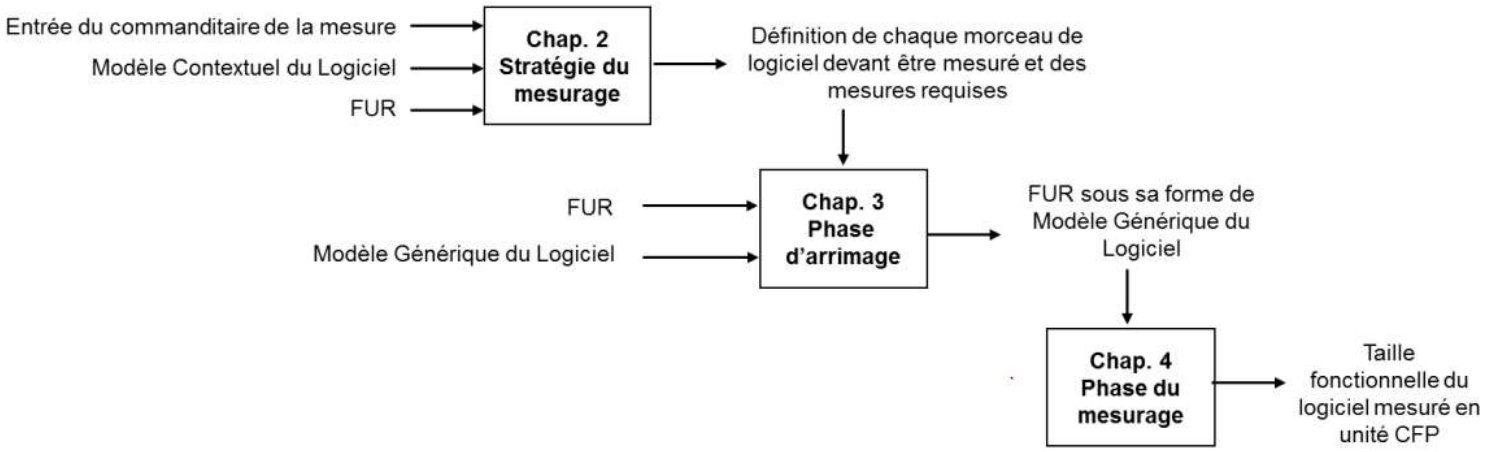


Figure 17 Aperçu de la méthode Cosmic [Cosmic, 2016]

6.1. La phase de la stratégie de mesurage

Cette phase permet de déterminer ce qui va être mesuré, pourquoi cette mesure est faite et à qui cette mesure est destinée. Cette phase est définie par cinq paramètres :

6.1.1. La raison d'être du mesurage

La définition de Cosmic sur la raison d'être du mesurage est la suivante : « *Un énoncé qui indique pourquoi un mesurage est exigé, et comment le résultat sera employé.* »

Il faut donc établir : pourquoi fait-on la mesure, pour qui, et dans quel but ? La raison d'être définit notamment les artéfacts mesurés et quand on fait le mesurage, avant le développement, au cours du développement pour estimer l'avancement ou après le développement pour estimer le rendement d'un développeur.

Ce paramètre est important car il sera déterminant pour les quatre autres paramètres.

6.1.2. Le périmètre du mesurage

De nouveau une définition de Cosmic sur le périmètre du mesurage est la suivante : « *Le périmètre du mesurage est l'ensemble des Fonctionnalités Utilisateur Requises (FUR) qui doivent être incluses dans une occurrence spécifique du mesurage de taille fonctionnelle.* »

La raison d'être aide à déterminer le périmètre car elle permet de déterminer :

- *Quel logiciel est inclus ou exclu du périmètre*
- *Et de quelle façon le logiciel inclus devrait être divisé en morceaux séparés, chacun avec son propre périmètre, à mesurer séparément.*

Le périmètre doit respecter deux règles :

- *Le périmètre de tout morceau de logiciel à mesurer doit être dérivé de la raison d'être du mesurage.*
- *Le périmètre de tout mesurage ne doit pas couvrir plus d'une couche du logiciel à mesurer*

Une couche est : « *Un partitionnement résultant d'une division fonctionnelle d'un système qui, ensemble, avec le matériel, forme un système informatique complet* »

6.1.3. Les utilisateurs fonctionnels

Cosmic définit les utilisateurs fonctionnels comme tels : « *C'est un (type d') utilisateur qui est un expéditeur et/ou un destinataire de données dans la Fonctionnalité Utilisateur Requise d'un morceau de logiciel.* »

De nouveau les utilisateurs fonctionnels sont tirés à partir de la raison d'être. Un utilisateur fonctionnel peut être un utilisateur du système mais aussi un autre système.

L'identification des utilisateurs fonctionnels permet ensuite d'identifier la frontière, celle-ci est située entre le logiciel à mesurer et l'utilisateur fonctionnel

Cette phase permet aussi d'identifier notre stockage persistant

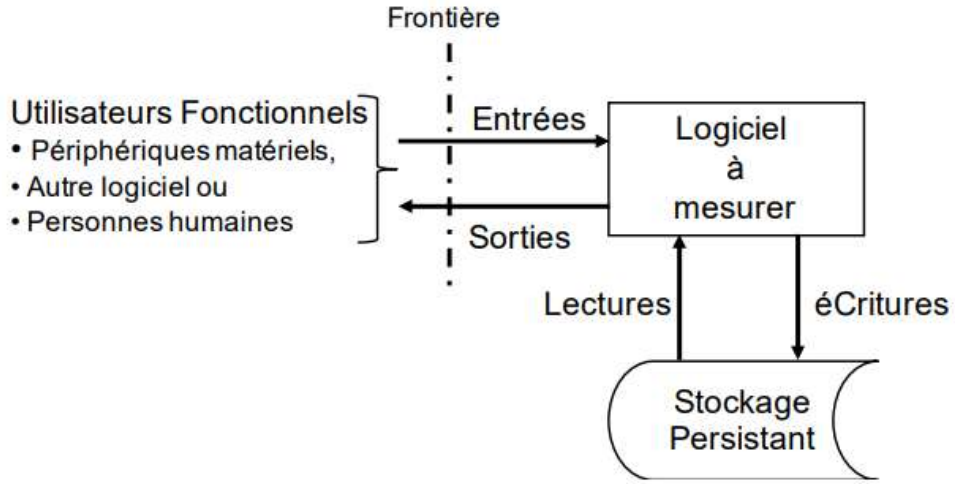


Figure 18 Utilisateur fonctionnel et frontière [Cosmic, 2016]

6.1.4. Le niveau de granularité

C'est le niveau de détail, on peut faire une analogie avec les cartes routières, on a les cartes à l'échelle d'un pays qui représentent les grands axes, et les cartes topographiques sur lesquelles on voit plus de détails comme l'altitude le dénivelé, …

6.2. La phase d'arrimage

Dans [Cosmic, 2016], on explique les concepts généraux de cette phase :

a) *Un évènement incite un **utilisateur fonctionnel** à demander un service au morceau de logiciel à mesurer. Un tel évènement est appelé un 'évènement déclencheur' et le service demandé un **'processus fonctionnel'**.*

b) *Les **processus fonctionnels** sont composés de deux types de sous-processus visant, soit à déplacer des données ("**mouvements de données**") ou encore à manipuler des données (**'manipulation de données'**). Les sous-processus de manipulation des données ne sont pas reconnus séparément, mais sont normalement pris en considération par les mouvements de données auxquels ils sont associés.*

c) *Un mouvement de données déplace un **'groupe de données'**. Un groupe de données est constitué d'attributs de données qui servent tous à décrire un 'objet d'intérêt', c'est-à-dire un objet qui est d'intérêt pour l'utilisateur fonctionnel concerné.*

d) *Il y a quatre types de mouvements de données **: Entrées** et **Sorties** qui déplacent un groupe de données depuis et vers un processus fonctionnel à travers la frontière vers/d'un utilisateur fonctionnel. **Lectures et éCritures** qui déplacent chaque groupe de données entre le processus fonctionnel et le stockage persistant.*

Dans la phase d'arrimage nous devons établir les différents processus fonctionnels, puis pour chaque processus fonctionnel les groupes de données qu'ils manipulent, et éventuellement leur attribut, puis enfin identifier tous les mouvements de données. Nous allons maintenant détailler ces étapes plus en détails.

6.2.1. Identifier les processus fonctionnels

Comme vu précédemment, avant de pouvoir déterminer les groupes de données et les mouvements de données nous devront tout d'abord identifier les processus fonctionnels.

Un processus fonctionnel est défini comme tel : « Un ensemble de mouvements de données représentant une partie élémentaire de la FUR du logiciel à mesurer, qui est unique dans cette FUR et qui peut être défini indépendamment de tout autre processus fonctionnel dans cette FUR. ». De plus, chaque processus fonctionnel est déclenché par un seul évènement déclencheur.Une fois les processus identifiés ceux-ci doivent répondre à ces 4 règles :

1. Un processus logiciel n'appartient qu'à une seule couche et est entièrement contenu dans un seul périmètre
2. Toute entrée ne déclenche qu'un seul processus fonctionnel
3. Un processus fonctionnel contient au moins deux mouvements de données : une entrée et une sortie ou écriture
4. Le processus est considéré comme terminé une fois qu'il a rempli les exigences de la FUR en réponse à son événement d'entrée.

6.2.2. Identifier les groupes de données

Avant de définir ce qu'est un groupe de données il faut tout d'abord définir ce qu'est un objet d'intérêt. Selon Cosmic un objet d'intérêt est « *Toute 'chose', dans le monde de l'utilisateur fonctionnel qui est identifiée du point de vue des Fonctionnalités Utilisateurs Requises (FUR) au sujet duquel le logiciel est requis d'effectuer un traitement et/ou stocker des données. Ce peut être une chose physique, aussi bien qu'un objet conceptuel ou partie d'un objet conceptuel* ».

Maintenant que le concept d'objet d'intérêt est défini nous pouvons définir le groupe de données qui est : « *Un ensemble, non vide, et non ordonné d'attributs de données où chaque attribut de donnée décrit un aspect complémentaire du même objet d'intérêt.* »

Si nous prenons un des exemples de Cosmic, dans les cas d'un processus de commande composé d'une commande et de lignes de commande, il y aurait deux objets d'intérêt, la commande et la ligne de commande, et deux groupes de données, les données de commande et les lignes de commande.

6.2.3. Identifier les attributs de données

Cette étape est optionnelle vu que dans Cosmic on compte les mouvements des groupes de données. Cependant dans le cas de la modification d'un logiciel, il est intéressant de connaitre les attributs des objets pour savoir quel mouvement de données doit être comptabilisé ou non. La définition d'attribut de donnée selon Cosmic est : « *La plus petite parcelle d'information, dans un groupe de données, possédant une signification dans la perspective des Fonctionnalités Utilisateurs Requises (FUR) du logiciel.* »

6.2.4. Identifier les mouvements de données

Il existe 4 types de mouvements de données : les entrées, les sorties, les lectures et les écritures.

Avant de définir ces concepts il faudrait tout d'abord définir ce qu'est un stockage persistant. Selon Cosmic un stockage persistant est un : « *Stockage qui permet à un processus fonctionnel de conserver un groupe de données au-delà de la durée du processus fonctionnel et/ou à partir duquel un processus fonctionnel peut récupérer un groupe de données conservé par un autre processus fonctionnel, conserver une occurrence antérieure du même processus fonctionnel ou conservé par un autre processus »*.

Cosmic introduit aussi la définition de manipulation des données, c'est : *« Tout ce qui survient aux données autre qu'un mouvement de données vers l'intérieur ou l'extérieur d'un processus fonctionnel, ou entre un processus fonctionnel et un stockage persistant* ». Selon Cosmic toutes manipulations de données peut être associé à un mouvement de données.

Il est donc important de distinguer utilisateur fonctionnel et stockage persistant, de plus le stockage persistant se trouve à l'intérieur de la frontière. Le concept d'entrée, sortie, lecture, écriture et stockage persistant a déjà été représenté dans la figure 18.

6.2.4.1. Entrée

Définition selon Cosmic : « *Un mouvement de données qui déplace, à travers la frontière, un groupe de données depuis un utilisateur fonctionnel dans le processus fonctionnel où il est requis »*

6.2.4.2. Sortie

Définition selon Cosmic : « *Un mouvement de données qui déplace, à travers la frontière, un groupe de données d'un processus fonctionnel vers l'utilisateur fonctionnel qui le requiert* ».

6.2.4.3. Lecture

Définition selon Cosmic : « *Un mouvement de données qui, déplace un groupe de données depuis son stockage persistant dans le processus fonctionnel qui le requiert* ».

6.2.4.4. Ecriture

Définition selon Cosmic : « *Un mouvement de données qui déplace un groupe de données depuis un processus fonctionnel vers le stockage persistant* ».

6.3. La phase de mesurage

Dans cette phase finale on applique l'unité de mesure Cosmic puis on applique la fonction de mesure. Appliquer l'unité de mesure consiste à attribuer 1 CFU[9] a chaque mouvement de données, appliquer la fonction de mesure consiste à additionner tous ces CFU.

Dans cette section de Cosmic on explique aussi comment on doit appliquer la méthode pour estimer la taille de modification d'un logiciel. Selon Cosmic une modification fonctionnelle consiste en : « *toute combinaison d'ajouts de nouveaux mouvements de données ou modifications ou suppressions de mouvements de données existants* ». Les modifications d'un logiciel peuvent être dues soit à l'ajout d'une FUR soit à la modification d'une FUR.

Si on mesure la taille d'un logiciel il ne faut donc comptabiliser que les mouvements de données qui ont été ajoutés, modifiés ou supprimés.

De plus, selon Cosmic il faut considérer qu'un mouvement de données est modifiés si son groupe de donnée est modifié, ou si cette manipulation de données a été modifiée. Un groupe de données est considéré comme modifié s'il y a au moins un attribut de données ajouté, supprimé ou modifié.

Finalement, en annexe Cosmic nous fournit une grille pour pouvoir relever la mesure. Dans cette grille chaque ligne correspond à des processus fonctionnels et les colonnes aux entries, exits, reads, writes et les totaux.

	Data Group Names												
Softw are Name A	Data Group 1	...	...	...	...	...	...	Data Group n	Entries	Exits	Reads	Writes	Total
Functional Process 1													
Functional Process 2													
Functional Process 3													
Functional Process 4													
Functional Process 5													
					Softw are A Totals								

Figure 19 Grille de mesurage [Cosmic, 2015]

[9] CFU : Cosmic Function Point, c'est l'unité de mesure utilisé par COSMIC

6.4. Modèle générique d'un logiciel selon Cosmic

En conclusion Cosmic résume un logiciel comme tel [Cosmic, 2016] :

a) *Un morceau de logiciel interagit avec ses utilisateurs fonctionnels au travers d'une **frontière**, et avec un **stockage persistant** à l'intérieur de cette frontière.*

b) *Les FUR d'un morceau de logiciel à mesurer peuvent être arrimées à des processus fonctionnels uniques.*

c) *Chaque processus fonctionnel est constitué de sous-processus.*

d) *Un sous-processus peut être soit un **mouvement de données** ou une **manipulation de données.***

e) *Il y a quatre types de mouvement de données**, Entrée, Sortie, éCriture et Lecture.** Une Entrée déplace un groupe de données dans un processus fonctionnel à partir d'un utilisateur fonctionnel. Une Sortie déplace un groupe de données d'un processus fonctionnel vers un utilisateur fonctionnel. Une éCriture déplace un groupe de données d'un processus fonctionnel vers un stockage persistant. Une Lecture déplace un groupe de données d'un stockage persistant vers un processus fonctionnel.*

f) *Un mouvement de données déplace un seul groupe de données.*

g) *Un groupe de données consiste en un ensemble unique **d'attributs de données** décrivant un seul **objet d'intérêt**.*

h) *Chaque processus fonctionnel est démarré par un mouvement de données **entrée-déclencheur.** Le groupe de données déplacé par l'entrée-déclencheur est généré par un utilisateur fonctionnel en réponse à un **évènement déclencheur.***

i) *Un processus fonctionnel doit comprendre au moins un mouvement de données d'Entrée et un mouvement de données éCriture ou Sortie, c'est-à-dire qu'il doit comprendre un minimum de deux mouvements de données. Il n'y a pas de limite supérieure pour le nombre de mouvements de données dans un processus fonctionnel.*

j) *En tant qu'approximation pour les objectifs du mesurage, les manipulations de données des sous-processus ne sont pas mesurées séparément ; la fonctionnalité de toute manipulation de données est censée d'être tenue en compte par le mouvement de données auquel elle est associée.*

6.5. Travaux sur BPMN et Cosmic

Nous allons maintenant voir dans cette section ce qui a déjà été fait pour mettre en relation Cosmic et BPMN et dans la section suivante nous verrons ce qui a déjà été fait sur Cosmic avec les ERPs.

L'article de [Marín, et al., 2012] propose une solution pour mesurer un diagramme BPMN au moyen de Cosmic. Tout comme la méthodologie Cosmic sa mesure est séparée en trois parties.

6.5.1.1. La phase de la stratégie de mesurage

Dans cet article il définit la stratégie de mesurage comme ceci :

- Le **but** du mesurage est la mesure de la taille fonctionnelle d'une spécification d'un logiciel au moyen d'un diagramme BPMN. Le « scope » correspond au diagramme BPMN.
- Le **niveau de granularité** dépend du niveau de détail du diagramme BPMN.
- Les **utilisateurs fonctionnels** correspondent à chaque utilisateur du système modélisé et aux rôles du diagramme BPMN.
- La **frontière** est définie entre un « pool » du diagramme BPMN et l'utilisateur fonctionnel.

6.5.1.2. La phase d'arrimage

Selon [Marín, et al., 2012] les processus fonctionnels sont les activités du diagramme BPMN. Mais si une activité est accédée par plusieurs utilisateurs fonctionnels on ne compte qu'un seul processus fonctionnel. Les groupes de données sont tous les « data object » qui sont soit en entrée ou en sortie de BPMN.

6.5.1.3. La phase de mesurage

L'auteur identifie les mouvements de données comme ceci :

- Identifier une entrée pour les « data objects » qui sont en entrée (data input)
- Identifier une sortie pour tous les « data objects » sortants (data output)
- Identifier une lecture à chaque fois qu'on consulte un « data object » dans la base de données.
- Identifier une écriture à chaque fois qu'on écrit un « data objet » dans la base de données.

Il faut ensuite additionner tous les mouvements de données d'une FUR afin d'avoir la taille fonctionnelle d'un processus, et additionner la taille de tous les processus pour connaitre la taille du logiciel.

6.6. Travaux sur Cosmic et les erp

Dans l'article de [Téllez, 2009], l'auteur propose d'estimer l'effort d'implémentation d'un ERP en modélisant celui-ci avec eEPC. EEPC est un autre langage de modélisation de processus qui peut être comparé à BPMN. Ensuite il applique Cosmic sur cette modélisation.

EEPC veut dire extended EPC, c'est une extention de EPC (Event-driven Process Chain) qui fournit en plus des « Organisational Unit » et des « information object »

EEPC est composé de :

- « Event » : Un événement déclenche une fonction ou peut être l'état de fin d'une activité. Ex. « commande arrivée ».
- « Function » : Une fonction représente une tâche qui doit être exécutée par une personne. La fonction a pour but d'avoir un état final et peut produire une sortie. Ex « préparer une commande »
- « Logical operators » : Permet de définir des points de décisions inclusifs ou exclusifs (OR,XOR), ou définir une exécution parallèle (AND) , comme les « Gateways » BPMN
- « Process Path » : c'est plus ou moins comme les « sub-process » BPMN, ils permettent d'établir une hiérarchie dans les processus, du plus abstrait au plus détaillé. Ex. processus EPC détaillé de « préparer une commande »
- « Control Flow »: ce sont les connecteurs qui relient les éléments précédents afin de déterminer la chronologie de ceux-ci
- Organization Unit : ils permettent de représenter des rôles ou personnes, qui sont responsables d'une fonction
- « Information object » : permet de représenter une information fournie à une fonction en entrée ou en sortie d'une fonction

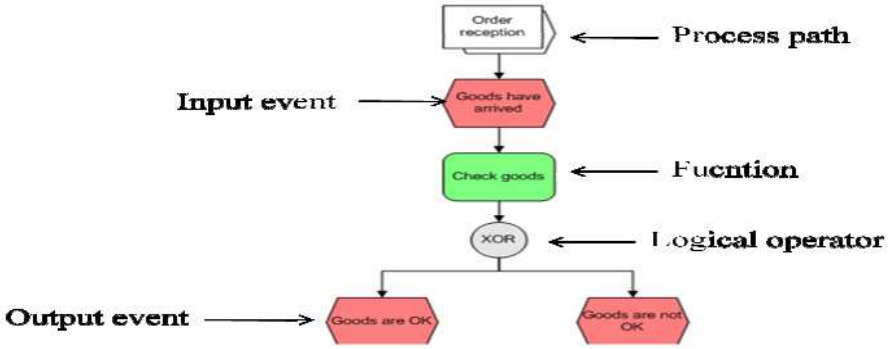


Figure 20 Exemple de EPC [Téllez, 2009]

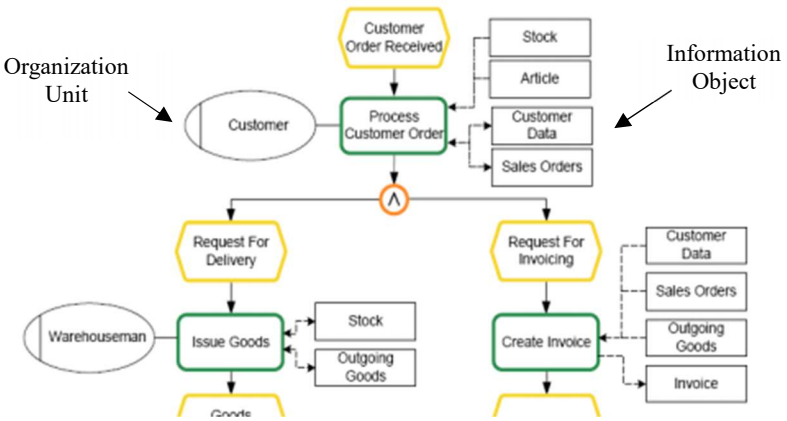


Figure 21 Exemple de eEPC [Téllez, 2009]

6.6.1.1. La phase de la stratégie de mesurage

- Définition du but : mesurer la taille fonctionnelle du nouvel ERP afin de pouvoir déterminer l'effort de développement
- Définition du périmètre de la mesure : correspond aux fonctionnalités d'un package ou les fonctionnalités d'un composant
- Le niveau de décomposition : il dépend des fonctionnalités qu'on a choisi de représenter
- Niveau de granularité : Ce sera une représentation haut niveau tout comme eEPC.
- Les utilisateurs fonctionnels: il faut considérer chaque unité d'organisation du diagramme eEPC comme un utilisateur du système.
- La frontière : il place la frontière entre les utilisateurs fonctionnels et le reste du diagramme EPC, cependant il faut exclure les process.

6.6.1.2. La phase d'arrimage

Identification des processus fonctionnels : il faut identifier un processus fonctionnel pour chaque processus représenté en eEPC. Il faut considérer tous les processus contenus dans un « process path » aussi.

Identification des « data groups » : il faut prendre toutes les « information object » comme groupes de données

6.6.1.3. La phase de mesurage

Identification des entrées :

- Il faut identifier une entrée pour chaque évènement d'entrée qui a été identifié comme groupe de données, mais uniquement si cet événement ne provient pas d'une autre fonction.
- Il faut aussi considérer une entrée quand un évènement amène un « data group » dans le processus

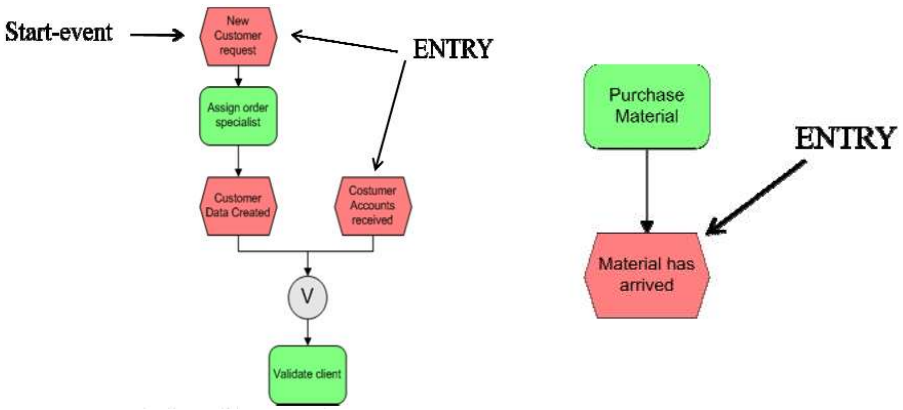


Figure 22 Exemple des entrées [Téllez, 2009]

Identification des sorties:

- Identifier une sortie pour chaque événement « sortant » que l'on n'utilise pas comme évènement d'entrée. Il faut les prendre comme sorties uniquement s'ils produisent un groupe de données.
- Prendre comme sortie chaque évènement qui ont un « information object » qui vient du système.

Identification des lectures :

- Identifier tous les « information object » qui sont lus par la fonction. Identifier comme lecture tous les « input event » qui servent de précondition

Identification des écritures :

- Identifier tous les « information object » qui sont écrits par la fonction.

Il suffit enfin de sommer toutes les entrées, sorties, lectures et écritures d'un processus pour connaitre la taille fonctionnelle d'un processus, puis sommer toutes les tailles des processus afin d'avoir la taille fonctionnelle du système.

Partie III
CONTRIBUTION

Chapitre 7: Exemple d'une modification de Odoo : ajouter un champ

Afin de réaliser comment Odoo peut être modifié, nous allons dans ce chapitre présenter une petite modification qui consistera à ajouter un champ "réduction" dans la fiche du client, et comment appliquer cette réduction à chaque vente destinée à ce client. Cet exemple sera aussi utilisé dans notre premier cas pratique. Dans cet exemple nous considérons que le module de vente et l'option réduction sont déjà installés. Nous allons donc séparer cette modification en deux parties, la première étant l'ajout du champ dans la fiche du client et la deuxième étant l'application de cette réduction à la vente.

7.1. Ajout d'un champ dans la fiche du client

Pour pouvoir rajouter un champ dans la fiche du client il faudra tout d'abord modifier l'objet « Partner » afin de lui ajouter un attribut. Ceci se fait dans le code python, il faut déclarer un nouvel objet (ligne 17), puis indiquer à l'ORM que cet objet hérite de Partner (ligne 18) et déclarer le nouveau champ (ligne 20-22), de plus nous mettons ce champ à 0 par défaut (ligne 24).

```
17  class res_partner(osv.osv):
18      _inherit = 'res.partner'
19
20      _columns = {
21          'discount_all': fields.float('Discount', help='Discount percentage'),
22      }
23      _defaults = {
24          'discount_all': 0,
25      }
26
```

Une fois le champ déclaré, il faudra modifier la vue qui affiche le formulaire du client afin d'y ajouter ce champ. Ceci se fait au moyen d'un fichier XML, nous déclarons donc une nouvelle vue, et nous faisons hériter celle-ci grâce à la ligne 8. Pour modifier la vue nous utilisons d'abord Xpath qui, comme dit dans le chapitre 2:, est un outil qui permet de rechercher un élément XML puis ensuite le modifier. Ici nous recherchons donc l'élément XML de type group dont le champ « name » est « date ». Ensuite nous rajoutons le champ (ligne 11), juste après le groupe récupéré.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <openerp>
3      <data>
4
5          <record model="ir.ui.view" id="res_unamur_discount">
6              <field name="name">res.partner.form</field>
7              <field name="model">res.partner</field>
8              <field name="inherit_id" ref="base.view_partner_form" />
9              <field name="arch" type="xml">
10                 <xpath expr="//group[@name='date']" position="after">
11                     <field name="discount_all" />
12                 </xpath>
13             </field>
14         </record>
15
16
17     </data>
18 </openerp>
19
```

7.2. Application de la réduction automatiquement

Un fois le champ "discount" ajouté dans la fiche client, nous pouvons maintenant implémenter la réduction automatique. Avant de s'y attaquer, il faut savoir que Odoo permet de créer des fonctions qui sont appelées à chaque fois qu'un champ est modifié dans l'interface graphique. Nous allons donc profiter de cette fonctionnalité pour détecter quand on rentre le nom du produit dans la commande et ensuite renvoyer la réduction à appliquer.

De nouveau cette modification se fait dans le code Python, tout comme pour l'ajout d'un champ nous commençons par créer un nouvel objet qui va hériter de l'objet que l'on veut modifier, ici « sale.order.line » en l'occurrence. Ensuite nous surchargeons la fonction qui nous intéresse, celle-ci est la fonction « onchange ».

Dans cette fonction nous commençons par appeler la fonction « super » (ligne 32) afin d'exécuter les « onchange » des fonctions de l'objet parent, et nous récupérons le résultat de cette méthode. Ensuite, nous allons récupérer la réduction à appliquer dans la base de données, pour ce faire nous récupérons d'abord le client dans la base de données à partir de son id, et nous connaissons l'id du client car quand le frontend appelle la méthode « onchange » celui-ci nous passe tous les champs qui ont déjà été remplis dans le formulaire.

Un fois le client récupéré nous n'avons qu'à renvoyer la réduction liée au client. Ceci est possible en rajoutant un nouvel élément au tableau des champs, cet élément doit avoir le même nom que le champ qu'on veut modifier, et l'interface graphique changera automatiquement les champs du formulaire à partir ce tableau.

```
28  class sale_order_line(osv.osv):
29      _inherit = 'sale.order.line'
30
31      def onchange(self, cr, uid, id,fields,dz,dzt, context=None):
32          res=  super(sale_order_line, self).onchange(cr, uid, id,fields,dz,dzt, context)
33
34
35          partnerOrm=self.pool.get('res.partner')
36          partner=partnerOrm.browse(cr,uid,[fields['order_id']['partner_id']],context)
37
38          values=res['value']
39          values['discount']=partner.discount_all
40          res['value']=values
41          return res
42
```

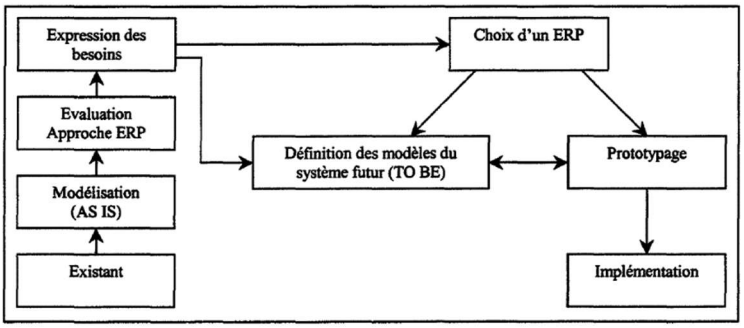
Chapitre 8: Critique des méthodologies existantes, et proposition d'une nouvelle méthodologie

8.1. Critique de [Mordant, 2007]

Le mémoire de [Mordant, 2007] est une bonne première approche pour la mise en œuvre d'un ERP. Les deux dernières parties de sa phase d'implémentation et sa phase opération sont similaires à une implémentation Odoo et à la gestion après-vente de Odoo. Cependant pour la partie définition du « business case » et alignement nous allons nous baser sur les thèses de Boutin et de Lacombe.

8.2. Critique de [Boutin, 2001]

Nous allons maintenant voir ce qui n'est pas compatible dans la méthodologie de P. Boutin avec nos objectifs et modifier cette méthodologie afin qu'elle corresponde à nos objectifs. Pour rappel la méthodologie de P. Boutin suit ce schéma.



Nous allons maintenant critiquer et modifier cette méthodologie dans les 6 points suivants :

8.2.1. La phase de modélisation, évaluation et expression des besoins

Nous avons choisi de fusionner ces trois étapes afin d'économiser du temps dans l'analyse des besoins. Ces trois étapes constitueront notre analyse des besoins qui exprime uniquement les besoins futurs de l'entreprise.

8.2.2. La phase de sélection de l'ERP

Nous avons déjà notre ERP, donc nous allons supprimer cette phase.

8.2.3. Ajout d'une phase qui consiste à élaborer un modèle de Odoo

Cette phase sera une phase qui est séparée du processus d'implémentation, elle consiste en la phase de « Domain engineering », c'est à dire rechercher de nouvelles fonctionnalités de Odoo dans la communauté et les incorporer dans notre modèle de Odoo pour des projets futurs.

8.2.4. La phase de définition du système futur

Dans cette phase nous sélectionnerons une configuration possible de Odoo à partir de notre modèle « as-wished » et des différentes configurations de Odoo déterminées dans la phase précédente

8.2.5. La phase de prototypage

Cette phase est réalisée si notre configuration de Odoo ne permet pas de remplir tous les besoins du modèle « as-wished ». Dans cette phase nous allons alors déterminer les changements à effectuer pour ensuite pouvoir estimer la taille de ces changements.

8.3. Critique de [Lacombe, 2015]

Pour rappel voici méthodologie de Lacombe :

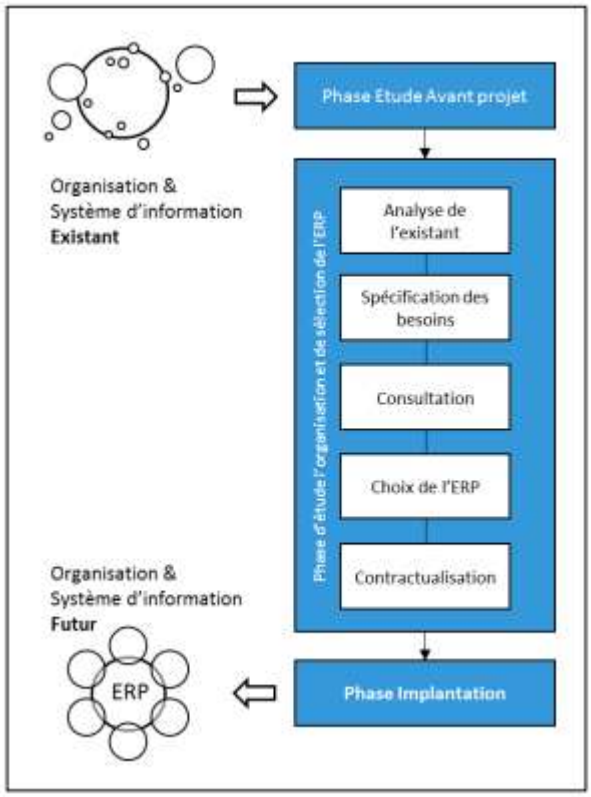


Figure 23 Méthodologie de [Lacombe, 2015]

Dans notre mémoire nous considérons que certains éléments de la phase d'étude avant-projet ont déjà été réalisés, cependant certains éléments de cette phase vont se retrouver dans notre phase de « domain engenering ».

Pour les deux premières parties nous allons les fusionner comme pour le point 8.2.1.

La phase de consultation et de choix de l'ERP seront supprimées.

Nous gardons la phase de contractualisation avant l'implémentation. En effet nous aurons toutes les cartes en main pour pouvoir estimer le coût de l'implémentation et ainsi proposer un contrat à notre client.

8.4. Proposition d'une nouvelle méthodologie

En plus des modifications citées dans les points précédents nous séparons les activités en deux processus qui correspondent au processus « Domain engineering » et au processus « Application engineering ». Toutes ces modifications aboutissent à une nouvelle méthodologie : (Ce diagramme est aussi disponible en annexe 1 pour plus de lisibilité)

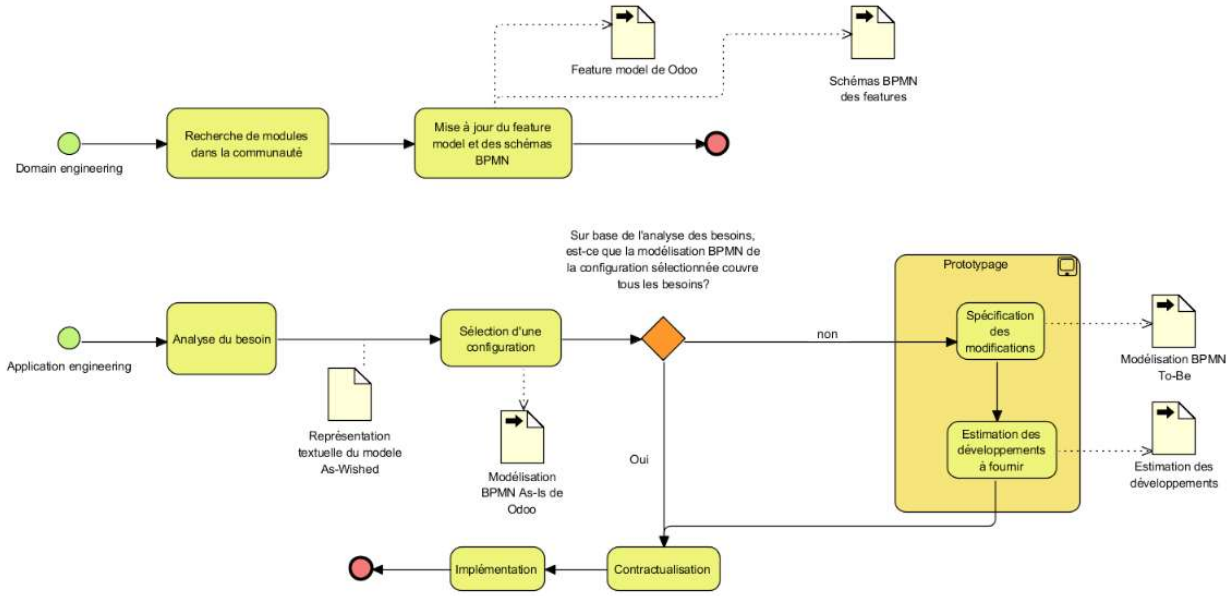


Figure 24 Notre méthodologie

8.4.1. Domain engineering

Notre processus de « Domain engineering » consiste à rechercher de nouvelles fonctionnalités à notre « Feature model », et ce processus est divisé en deux tâches.

La première est la recherche de modules, c'est à dire les installer et constater les changements qu'ils apportent à Odoo.

La deuxième consiste à partir des observations que l'on a faites à l'étape précédente, à modéliser ces modifications en BPMN et mettre à jour le « Feature model » de Odoo.

Une fois la deuxième tâche terminée ce processus peut être recommencé pour retrouver encore d'autres nouvelles fonctionnalités.

8.4.1.1. Recherche de modules dans la communauté

Pour éviter de se lancer dans trop de développement personnalisé dans le point 8.4.2.4, il serait intéressant de voir s'il n'existe pas dans la communauté des modules qui couvriraient des fonctionnalités pour des projets futurs. Pour ce faire il faut rechercher sur le site de Odoo des modules qui pourraient apporter des solutions intéressantes pour des projets futurs. Pour voir si un module est utile il faut lire sa description, ou à défaut d'une description complète l'installer sur une installation de test afin de voir comment celui-ci fonctionne. Il n'y a pas vraiment de méthode définie pour cette partie.

8.4.1.2. Mise à jour du Feature model et des diagrammes BPMN

Si on trouve des modules qui apportent des modifications intéressantes, il faut les rajouter à notre « Feature Model » afin de l'améliorer et ainsi pouvoir proposer ces nouvelles fonctionnalités à de futurs projets. Plus on effectue de projets Odoo, plus notre « Feature Model » grandit et devient complet. Chaque nouvelle fonctionnalité doit être documentée avec un diagramme BPMN.

8.4.2. Application engineering

Dans cette phase nous allons élaborer la meilleure solution pour pouvoir implémenter l'ERP Odoo. Le but de cette phase est de trouver une solution qui conviendra au mieux au client tout en s'assurant que celle-ci nécessitera le moins de développement possible (et donc le plus de réutilisations possibles)

8.4.2.1. Analyse du besoin

Afin de réaliser les étapes suivantes il faut avoir une analyse du besoin. Cette analyse du besoin permet de cerner ce que l'ERP doit prendre en charge, les processus qu'il va couvrir, comment se déroulent ces processus et quelles données doivent être traitées. Cette analyse représente notre modèle « As-Wished »

Il peut aussi être intéressant de lister les données que certains processus manipulent afin de voir si tous les attributs des données manipulées sont déjà pris en charge par Odoo ou s'il faudra modifier Odoo pour les y inclure. Cela permettra aussi de planifier la migration des données d'un éventuel système déjà existant dans l'entreprise.

8.4.2.2. Sélection d'une configuration

Cette étape consiste à parcourir le Feature Model de Odoo afin de choisir les fonctionnalités (feature) qui couvriront le plus les besoins du client. Cette sélection se fait grâce à la modélisation des besoins du client.

Une fois la sélection de modules faite, il faut représenter cette sélection en un diagramme BPMN. Ceci est fait en regroupant tous les diagrammes BPMN des « features » sélectionnés dans notre Feature model. La compatibilité de ces diagrammes est assurée par les contraintes du Feature model.

8.4.2.3. Test « Tous les besoins sont-ils couverts ? »

Maintenant que nous avons à la fois une modélisation des processus du client et la modélisation BPMN de notre configuration de Odoo, nous pouvons répondre à cette question :

« Sur base de l'analyse des besoins, est-ce que la modélisation BPMN de la configuration sélectionnée couvre tous les besoins ? »

La réponse à cette question est trouvée en comparant le modèle « As-Wished » au modèle « As-is » de Odoo afin de voir ce qu'Odoo ne peut pas prendre en charge et donc quelle fonctionnalité il faut implémenter nous-même.

Si aucune fonctionnalité ne manque, c'est qu'Odoo peut être utilisé tel quel pour l'entreprise sans devoir faire de recherche de modules ou de développements.

Mais ce n'est généralement pas le cas, il nous faut donc trouver une solution pour combler ces manques, ceci peut être fait soit en refaisant du « Domain engineering » cette à dire effectuer une recherche de module dans la communauté ou en développant nous-même ce module (ou confier ce développement à une entreprise spécialisée).

Si des fonctionnalités sont en trop il faut soit voir si cela ne dérange pas le client, sinon passer au point suivant pour essayer de trouver une solution.

8.4.2.4. Prototypage

Dans cette phase nous allons à partir de la représentation BPMN de la configuration choisie, et modifier celle-ci. Ensuite, nous allons mesurer ces modifications en vue de l'élaboration d'un devis pour accord du client. Suite à la phase de prototypage on pourrait aussi découvrir qu'il ne sera pas possible de réaliser ce projet avec Odoo.

8.4.2.5. Contractualisation

Une fois qu'on sait quelles fonctionnalités devront être mises en place et quels modules devront être créés et quelle est la taille de ces modules on peut proposer un devis au client en vue d'une éventuelle contractualisation. Ce contrat définira notamment si l'ERP doit être hébergé par le client ou l'entreprise, et quelles offres de maintenance sont couvertes par ce contrat.

8.4.2.6. Implémentation

Cette étape n'est pas couverte par ce mémoire, mais nous allons tout de même lister les tâches que cette étape devrait couvrir. Ces tâches sont similaires aux dernières étapes du mémoire de [Mordant, 2007]. L'implémentation doit donc contenir le développement des modules spécifiés dans notre étape de prototypage,

Ensuite en fonction du contrat, l'installation soit d'un serveur Odoo chez le client, ou bien l'installation d'une nouvelle instance sur le serveur de l'entreprise.

Puis l'installation et la configuration des modules sélectionnés et développés. L'importation des données d'un ancien système éventuel.

Et pour finir, mener à bien les tests post-implémentation, suivis de la mise en production et des maintenances prévues selon le contrat.

Chapitre 9: Proposition d'une modélisation de Odoo

9.1. Feature model de Odoo

Après l'étude des modules de base de Odoo, nous avons réalisé le « Feature model » suivant :

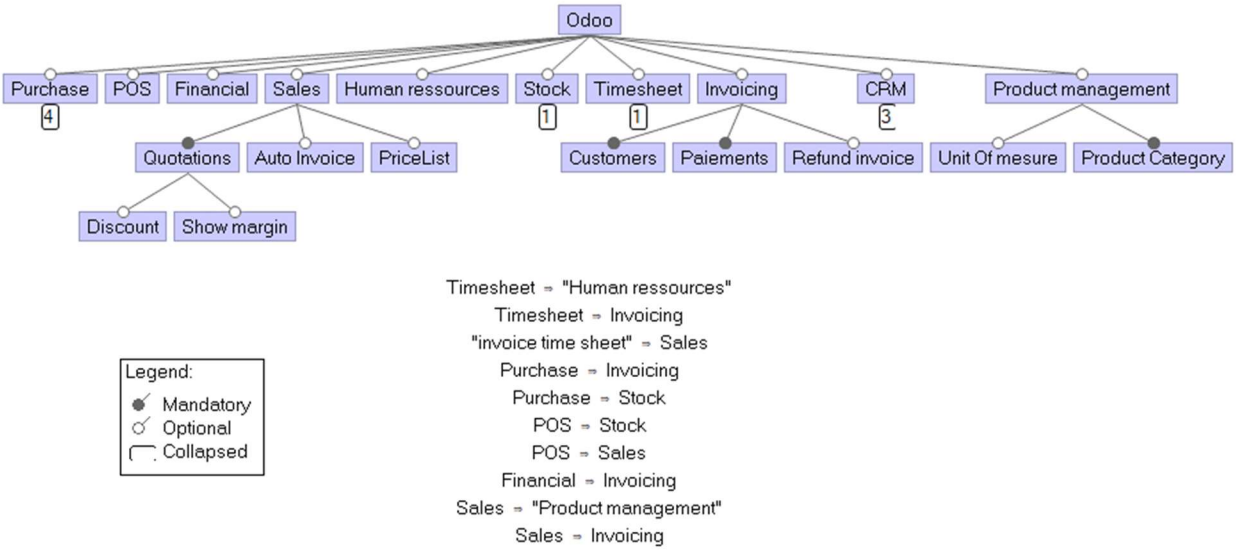


Figure 25 Feature model de Odoo

Cette modélisation est réalisée grâce au logiciel « FeatureIDE[10] » à partir d'une observation du comportement de Odoo après chaque installation d'un module. Par visibilité nous avons réduit l'affichage des features « Purchase », « Timesheet » et « CRM », le Feature model complet peut être trouvé dans l'annexe 3

9.2. Sélection d'une configuration

Ceci se fait à nouveau dans l'outil « FeatureIDE ». La création d'une configuration se fait en cochant les cases des « Feature » que l'on désire pour notre installation et le programme résout les dépendances et s'assure que notre configuration est valide. Des exemples de sélection de configurations sont disponibles dans la quatrième partie de ce mémoire.

[10] FeatureIDE : http://wwwiti.cs.uni-magdeburg.de/iti_db/research/featureide/

9.3. Représentation BPMN de Odoo

Avant de présenter notre modélisation de Odoo nous allons nous intéresser à la relation entre notre Feature model et la représentation BPMN de Odoo.

On peut trouver dans la littérature des travaux ([Montero, et al., 2008], [Schnieders, et al., 2006]) qui font un lien entre Feature model et BPMN. Le premier article propose une traduction automatique de Feature model en BPMN, mais cette traduction automatique n'est pas pertinente dans notre contexte. La deuxième publication utilise des annotations sur les diagramme BPMN afin d'identifier les points de variation et les variantes, nous nous sommes donc plus intéressés à cette méthode.

Nous allons nous inspirer du document de [Schnieders, et al., 2006], pour proposer une modélisation BPMN d'une configuration d'un « Feature model » en BPMN.

Pour notre modélisation de Odoo nous avons un processus par feature du premier niveau de notre feature model (CRM, Sales, Invoicing, …). Ensuite, pour chaque nouvelle feature, nous reprenons une copie du diagramme du feature parent et nous faisons nos modifications sur cette copie. Trois modifications sont possibles :

- L'ajout d'éléments
- La suppression d'éléments
- La modification d'éléments

Pour l'ajout d'éléments, cet ajout peut se faire avec des tâches, des points de choix, des « data objects » ou des flux (séquence flow) dans ce cas tous les éléments ajoutés doivent être dans un cadre noir, ce cadre sera annoté avec la mention « Add » suivi du nom du feature. Un exemple se trouve dans la figure 26 entre la tâche « confirm sale » et l'évènement « View Invoice », cet exemple est tiré du diagramme de l'annexe 4.3.

Pour la suppression d'éléments, celle-ci peut se faire sur les tâches, des points de choix, des data object ou des flux (séquence flow) dans ce cas tous les éléments supprimés doivent être dans un cadre rouge, ce cadre sera annoté avec la mention « Delete » suivi du nom du feature. Un exemple se trouve de nouveau dans la figure 26 entre la tâche « confirm sale » et l'événement « View Invoice », cet exemple est tiré du diagramme de l'annexe 4.3.

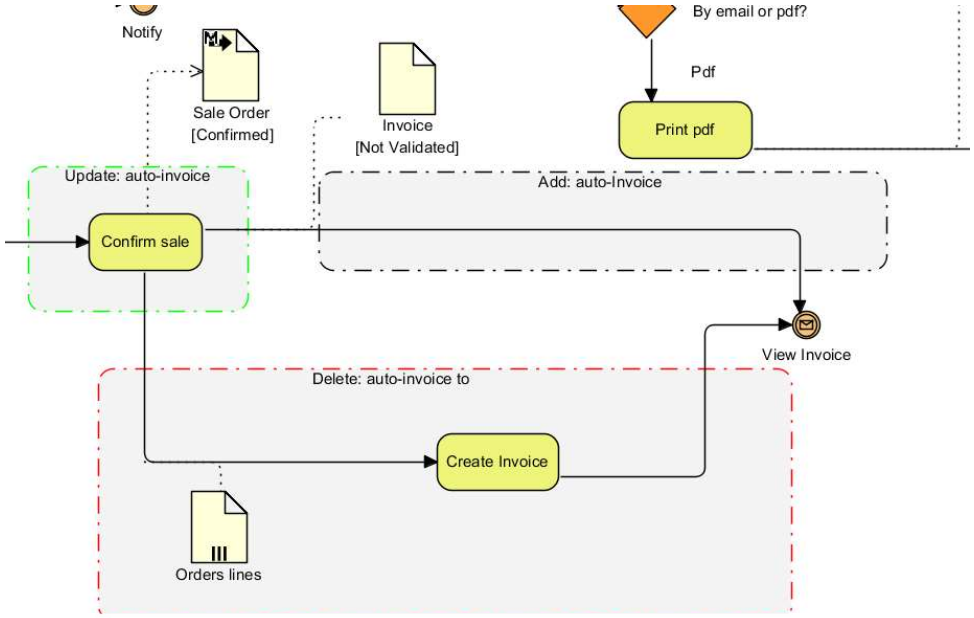


Figure 26 Exemple ajout et suppression

Dans le cas de la modification, celle-ci est applicable à un point de choix, une tâche ou un « data object ». Tous les éléments modifiés doivent se trouver dans un rectangle vert, ce cadre sera annoté avec la mention « Update » suivi du nom du feature. Un exemple peut être trouvé à la figure 27, cet exemple est tiré du diagramme de l'annexe 4.2.1.

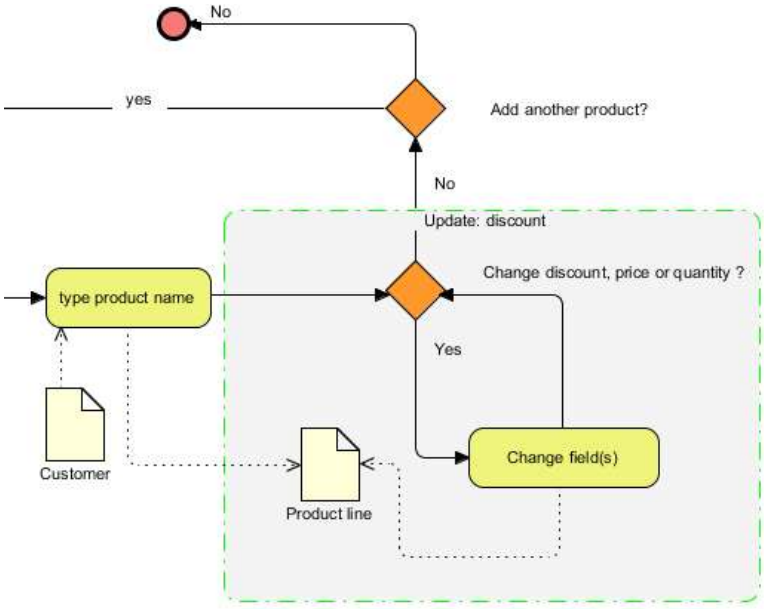


Figure 27 Exemple modification

Les diagrammes des feature « Sales », « Invoicing », « Stock » et « Products Management » ont leur diagramme BPMN en annexe 4.

9.4. Description des « data objects »

Dans le point 6.2.3, nous avons besoin de connaître les attributs des « data object » manipulés. Pour ce faire, nous dressons un tableau décrivant chaque attribut du « data object » comme tel :

- Son nom,
- Son type,
- Si sa valeur est obligatoire
- Optionnellement sa valeur par défaut.

De plus si l'objet a un champ state, on liste les états possibles du champ. Vous trouverez un exemple tiré de l'annexe 5.4 dans la figure 28.

Name	Type	Default	Mandatory?
State	State	Draft Quotation	yes
Date	Date	Now	yes
isPaid	Boolean	false	yes
Payment term	Payment Term		
Fiscal Position	Fiscal Position		
SalesPerson	Customer		
Order Lines	Multi Order Lines		yes

States: Draft Quotation, Quotation Sent, Sales Order, Sale to Invoice, Done

Figure 28 Exemple de description d'un « data object »

Chapitre 10: Prototypage

10.1. Spécification des modifications

Dans le cas où un développement personnalisé est nécessaire, nous modifions la modélisation BPMN « As-Is » de Odoo (modèle de la section 9.3) afin de correspondre au maximum aux désirs (modélisation « As-wished ») du client tout en réduisant le plus possible les développements à produire.

Les développements doivent être les plus minimes possibles car c'est l'activité la plus couteuse. De plus, avoir un logiciel différent de Odoo standard sera plus couteux à maintenir car à chaque mise à jour, les modifications pourraient ne plus être compatibles, et donc, de nouveaux développements pourraient être nécessaires. De plus ce code sera peut-être moins fiable, car moins testé que le code de la communauté.

Les modifications doivent être annotées afin de savoir quel processus doit être comptabilisé ou non dans la phase de mesure, les types de modifications possibles sont l'ajout, la modification et la suppression, et il faut suivre les mêmes conventions vues au point 9.3 pour annoter les diagrammes BPMN.

Si nous reprenons notre exemple du chapitre 7, nous avons les modifications suivantes pour la fiche client et pour la commande.

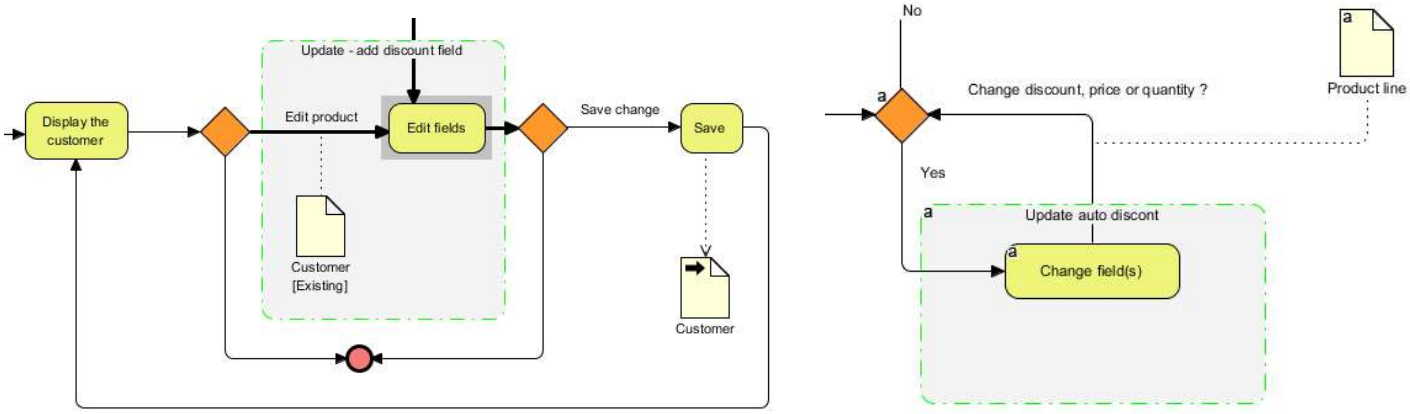


Figure 29 Modifications du chapitre 7

Ces modifications peuvent être fournies à l'équipe de « domain engineering » pour des travaux projets futurs.

10.2. Estimation des développements à effectuer

Une fois les développements modélisés nous pouvons estimer leurs tailles. Pour ce faire nous nous sommes inspiré des méthodes de [Marín, et al., 2012] et [Téllez, 2009] pour les appliquer à notre méthodologie.

Tout comme dans la méthodologie Cosmic notre mesure se fera en trois parties : la phase de stratégie, la phase d'arrimage et la phase de mesurage

10.2.1. La phase de stratégie

Dans cette partie nous allons surtout nous inspirer de [Téllez, 2009] :

- Définition du but : mesurer la taille fonctionnelle des modifications de l'ERP Odoo afin de pouvoir déterminer l'effort de développement des modules.
- Définition du périmètre de la mesure : les processus qui ont été modifiés dans la phase « Spécification des modifications ».
- Niveau de granularité : Le niveau sera le même niveau de détail que le diagramme BPMN.
- Utilisateurs fonctionnels : Pour les utilisateurs fonctionnels nous nous référons à l'article de [Marín, et al., 2012], ce sont les utilisateurs du système.
- La frontière : on la garde entre les utilisateurs du système et les processus modélisés.

Remarque : la phase de stratégie sera toujours la même pour nos cas pratiques.

10.2.2. La phase d'arrimage

Identification des processus fonctionnels : ce sont toutes les activités qui ont été modifiées

Identification des groupes de données : Ce sont nos « data object » et les événements du diagramme BPMN

Identification des mouvements de données :

Pour les mouvements de données nous allons établir les règles suivantes :

- Chaque tâche est provoquée par une action de l'utilisateur, nous allons donc compter une entrée.
- Chaque événement intermédiaire représente une sortie, en effet un événement pourrait être considéré comme une information (message d'erreur, changement de fenêtre) que le serveur Odoo nous renvoie.
- Pour les lectures nous pouvons les identifier par les lectures des « data objects », c'est à dire quand une flèche va d'un « data object » à un élément.
- De même pour les écritures, nous pouvons les identifier par les écritures des « data objects », c'est à dire quand une flèche va d'une activité à un « data object ».
- Les flèches qui pointent vers un point de choix ne rajoutent ni une entrée ni une sortie. Mais en général, l'ajout d'une « gateway » résulte en plusieurs appels à des activités ou évènements, donc des entrées ou sorties seront comptabilisées pour chacune de ces flèches. Une « gateway » pourrait aussi avoir besoin de lire des « data objects » pour prendre une décision, il faudra donc identifier ces lectures.

Tous ces mouvements de données sont illustrés à la page suivante dans la figure 30.

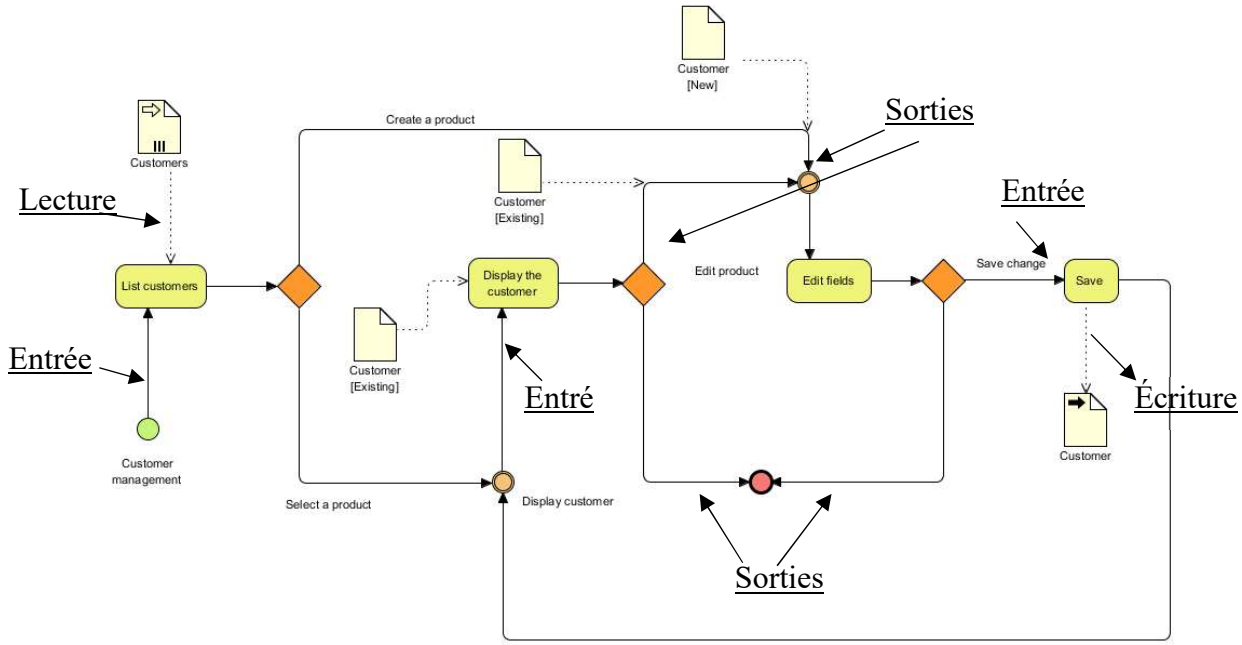


Figure 30 Exemple de mouvements de données

10.2.3. La phase de mesurage

Une fois les mouvements de données identifiés nous allons maintenant déterminer quels mouvements doivent être comptabilisés lors d'une modification. Comme vu au point 10.1 il y a trois types de modifications possibles : l'ajout, la suppression et la modification. Nous allons maintenant voir cela en détail.

10.2.3.1. Ajout d'éléments

Pour les ajouts d'éléments nous avons les règles suivantes :

- L'ajout d'une tâche implique une entrée
- L'ajout d'une flèche (sequence flow) résultera soit en une entrée ou en une sortie. Si cette flèche pointe un évènement, cela rajoutera une sortie, si elle pointe une tâche celle-ci rajoutera une entrée.
- L'ajout d'un « data object » ajoutera soit une lecture ou une écriture en fonction de la direction de la flèche (data association), on compte une écriture si la flèche pointe le « data object », et on compte une lecture si la flèche provient d'un « data object »
- Comme mentionné au point 10.2.2, l'ajout d'une « gateway » ne rajoute pas de point mais elle rajoutera surement des « flows » qui eux compteront pour des points. La « gateway » pourra aussi éventuellement lire des « data objets », dans ce cas on rajoutera une lecture par « data object ».
- L'ajout d'événements intermédiaires ou de connexions vers l'événement de fin rajoutera une sortie.

10.2.3.2. Suppression d'éléments

Pour les suppressions, Cosmic indique que la taille fonctionnelle de la suppression équivaut à la taille fonctionnelle des éléments supprimés du système, il faut donc utiliser les mêmes règles pour la suppression que pour l'ajout. C'est à dire :

- La suppression d'une tâche coutera les mêmes points d'entrée que son ajout.
- La suppression d'une flèche (flow) résultera aussi en une entrée ou une sortie, en fonction de l'élément vers lequel cette flèche pointait.
- La suppression d'un « data object » ajoutera soit une lecture ou une écriture en fonction de la direction de la flèche, on compte une écriture si la flèche pointe le « data object », et on compte une lecture dans le cas où la flèche provient d'un « data object ».
- La suppression d'une « gateway » comptera pour les points que celle-ci comptait en entrée, sortie et lecture.
- La suppression d'événements intermédiaires ou de flèches vers des événements de fin rajoutera des sorties.

10.2.3.3. Modification d'éléments

Et enfin, pour les modifications d'élément nous appliquerons ces règles :

- Si on modifie une tâche on compte son point d'entrée.
- Si un « data object » a été modifié, il faut uniquement compter une lecture ou une écriture si on accède à l'attribut qui a été modifié.
- Si on modifie une « gateway » cela dépendra des flux qui seront ajoutés ou supprimés et si les « data object » utilisés sont modifiés.

10.2.4. Rapport entre les points Cosmic et les modifications de Odoo

Nous allons maintenant brièvement faire une analogie entre les CFUs identifiés et les modifications de Odoo.

Compter un point d'entrée par tâche se justifie par le fait que pour ajouter une tâche il faudra surement rajouter des vues dans le frontend et donc modifier des fichiers XML.

Pour les points de sortie, chaque événement BPMN résultera en l'appel d'une action Odoo, ce qui implique l'ajout de code XML ou python.

Pour les lectures et écritures, ces points représenteront les interactions avec la base de données.

10.2.5. Grille Cosmic

Nous allons aussi nous inspirer de la grille Cosmic du point 6.3. Nous avons enlevé la liste de « data group » pour directement compter les points de chaque FUR. Un exemple de cette grille peut être trouvé en annexe 6.4.

Partie IV
APPLICATION

Dans notre méthodologie nous allons nous concentrer sur la partie « Application engineering » dans les deux premiers cas, et nous présenterons la partie « domain engineering » dans notre troisième cas. Nous ne détaillerons pas la partie conception étant donné que cette partie ne fait pas partie de ce mémoire. Pour rappel notre méthodologie peut être résumée par ce diagramme :

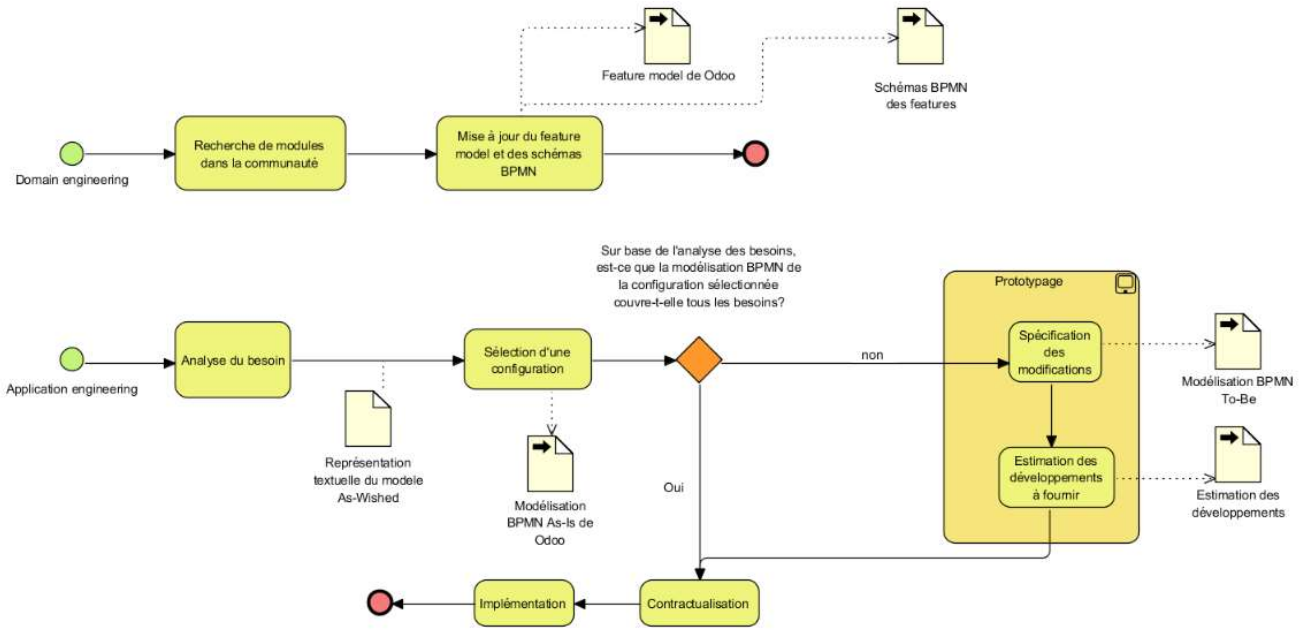


Figure 31 Notre méthodologie

Nous allons voir à présent 3 cas :

- Customer discount
- Commandes récurrentes
- Product Pack

Les phases abordées les parties seront :

1. Analyse du besoin
2. Sélection d'une configuration
3. Test : « Est-ce que tous les besoins sont couverts ? »
4. Prototypage
5. Spécification des modifications
6. Estimation des développements à effectuer
7. Contractualisation

La troisième partie comporte en plus les deux étapes du « domain engineering » :

1. Recherche de modules dans la communauté
2. Mise à jour du feature model et des schémas BPMN

Chapitre 11: Premier cas – Customer discount

11.1. Analyse du besoin

Un commerçant souhaite avoir un système de vente qui permette d'avoir un champ « réduction » dans la fiche de ses clients. Lors d'une commande le logiciel doit appliquer cette réduction si elle est mentionnée dans la fiche client.

11.2. Sélection d'une configuration

Suite à l'analyse du cas nous décidons d'installer le module « Sales » et le module « Discount ». Les modules « Invoicing » et « ProductMgmt » seront installés automatiquement car ils dépendent du module « Sales ».

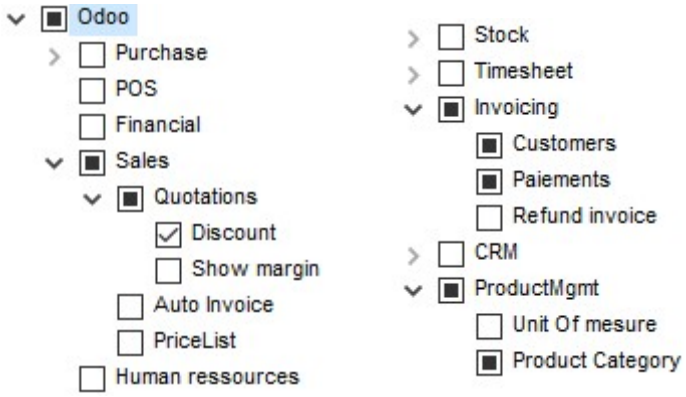


Figure 32 Configuration pour l'exemple 1

11.3. Test « Est-ce que tous les besoins sont couverts » ?

« Sur base de l'analyse des besoins, est-ce que la modélisation BPMN de la configuration sélectionnée couvre tous les besoins ? »

Non toutes les fonctionnalités ne sont pas couvertes, il manque la possibilité d'accorder la réduction automatique. Nous passons donc à la phase de prototypage.

11.4. Prototypage

11.4.1. Spécification des modifications

Nous devons donc faire des modifications à la fois dans la fiche du client et dans le système de commande.

Commençons par la fiche client, le diagramme modifié se retrouve en annexe 6.1. Dans ce diagramme nous avons donc modifié l'édition du formulaire et le « data object » « Customer », la description modifiée du « data object » Customer se trouve en annexe 6.2.

Pour la partie de gestion de la commande nous devons surcharger la tâche où l'on rentre le nom du produit, cette modification implique l'ajout d'une lecture vers le client.

11.4.2. Estimation des développements à effectuer

Il nous reste donc à mesurer la modification à faire, pour cela nous utiliserons la feuille Excel de l'annexe 6.4.

Les processus fonctionnels mesurées sont « Edit fields » et « Type product Name ».

Pour l'édition du client, nous avons donc une entrée pour la modification de la tâche, et une écriture pour la modification du « data object ».

Pour la modification du devis, nous avons aussi une entrée pour la modification de la tâche, et une lecture pour la lecture du client de la commande.

Toutes ces modifications nous font donc un total de 4 CFU

11.5. Contractualisation

Maintenant que nous savons quels modules il faudra installer, quelles modifications devront être faites et l'estimation des ces modifications qui est de 4CFU. Nous pouvons donc conclure quel est le prix à demander.

Chapitre 12: Deuxième cas – Commandes récurrentes

12.1. Analyse du besoin

1. Un client souhaite un système de vente, mais il ne veut pas une gestion du stock car il le gère à la main.
2. Il a une série de clients réguliers auxquels il accorde des prix réduits définis à l'avance (une liste de prix publics et une liste pour certains clients)
3. Pour tout achat de plus de 10 articles, il accorde une réduction de 2 % sur tous les articles suivants de la commande
4. Il aimerait aussi que ses factures soient payées en une seule fois, de plus il n'accorde jamais de remboursement de facture car si un produit est défectueux il le remplace.
5. Il désire avoir un moyen de pouvoir dupliquer des commandes récurrentes pour des clients qui viennent souvent acheter la même chose.

12.2. Sélection d'une configuration

Suite à la lecture de l'analyse, nous déduisons l'installation du module « Sales » pour remplir le point 1, l'activation du module « discount » pour remplir le point 3, et l'activation du module « pricelist ». Le module « pricelist » permet de pouvoir accorder des prix réduit pour les clients privilégiés ce qui correspond au point 2.

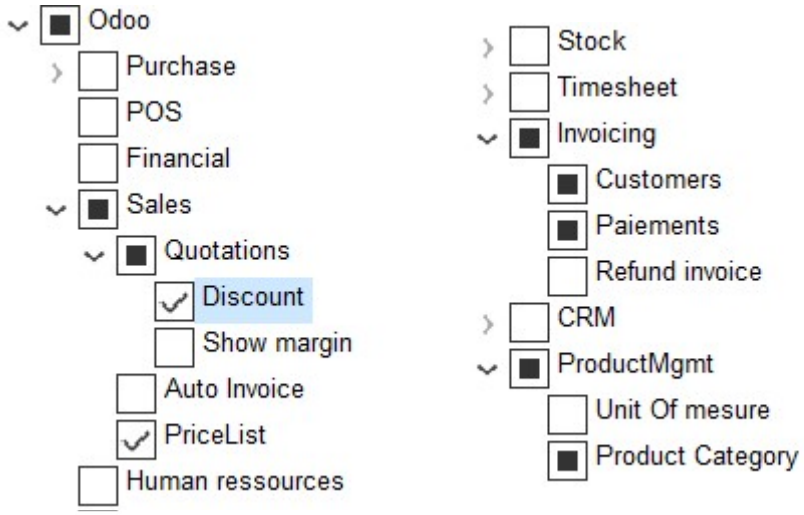


Figure 33 Configuration pour l'exemple 2

12.3. Test « Est-ce que tous les besoins sont couverts » ?

« Sur base de l'analyse des besoins, est-ce que la modélisation BPMN de la configuration sélectionnée couvre tous les besoins ? »

Les points 1 et 2 sont entièrement couverts. Pour le point 3 l'installation du module permet d'avoir des réductions, mais il faudra modifier Odoo pour attribuer cette modification automatiquement. Pour le point 4 ce sont aussi des modifications mineures. Le point 5 est à développer entièrement.

12.4. Prototypage

12.4.1. Spécification des modifications

Pour le point 3 il suffit qu'à chaque fois qu'on ajoute un produit, on vérifie le nombre d'articles et s'il y a plus de 10 articles, il faut mentionner qu'il y a 2 % de réduction. Cette modélisation est en annexe 7.

Pour le point 4 il suffit de surcharger la validation d'un paiement pour s'assurer qu'on paye la totalité de la commande en une fois. Pour ne pas permettre le remboursement d'une facture on masque tout simplement le bouton qui permet de créer le remboursement de la facture. Voir annexe 7.2

Pour le point 6 il n'existe pas de module similaire, nous allons devoir développer cette fonctionnalité entièrement, Pour ce faire nous implémentons :

1. Un bouton pour accéder au formulaire des commandes récurrentes, en plus d'une fenêtre contenant la liste de ces commandes (evénement Recurrent Orders et tâche list Orders)
2. Un formulaire permettant de rajouter une nouvelle commande récurrente, contenant un champ avec le nom du client, une référence et une liste d'articles. Cette liste d'articles contiendra pour chaque ligne le produit et la quantité désirée, le prix sera calculé en fonction du prix au moment de la commande.
3. Il faudra aussi ajouter un bouton qui permet de convertir la commande récurrente en commande.

Ce qui nous donne le diagramme en annexe 7.3

12.4.2. Estimation des développements à effectuer

Il nous reste à comptabiliser les points afin d'estimer la taille du développement.

Les processus fonctionnels mesurées sont « Type product Name » pour le modeule « Quotation », « Register paiment » et « Refund » pour le module « Invoicing », « List order », « Create order », « Add new order », « Types customer name », « add product name and qty » et « Save » pour le module « commande récurrente ».

12.4.2.1. Module Quotaion

Pour le module « Quotation » nous n'avons qu'**une** seule **entrée** qui correspond à la modification de la tâche « type product name ».

12.4.2.2. Module Invoicing

Pour le module « invoicing » nous avons **une entrée** qui correspond à la modification de la tâche « register payment », cette modification rajoutera aussi **une lecture**. Pour la suppression de l'événement « Refund » nous rajoutons **une écriture.**

Ceci nous fait un total de 3 CFUs pour ce module.

12.4.2.3. Nouveau module commande récurrente

Il nous reste à comptabiliser les points afin d'estimer la taille du développement du module commande récurrente.

Nous avons **une entrée** pour la création de la tâche « list orders », nous avons aussi **une lecture** qui correspond à la flèche entre « Recurrent orders » et « list orders ».

Pour la « Gateway » nous avons **trois sorties** qui correspondent aux évènements de fin, créer une nouvelle commande récurrente, et créer une nouvelle commande à partir d'une commande récurrente.

Pour la tâche « create order » nous avons **une entrée** et **une lecture** pour lire la commande récurrente et **une écriture** pour la commande et pour finir **une sortie** pour l'événement « View order ».

Pour l'ajout d'une commande récurrente, nous avons **une entrée** et **une lecture**, et ensuite, nous avons de nouveau **une entrée** pour la tâche « Type customer name,… » et **une sortie** pour l'évènement « add line ». Pour la tâche « Add product name … » nous avons **une entrée**, **une lecture** et **une écriture.** Et enfin, nous avons **une entrée** pour la tâche « save », qui entraine **une écriture** et **une sortie** pour l'événement « list recurrent order ».

Tout ceci nous fait un total de 19 CFUs pour ce nouveau module.

12.5. Contractualisation

Maintenant nous savons quels modules il faudra installer et quelles modifications doivent être faites et leur estimation qui est de 23 CFU. Nous pouvons donc conclure quel est le prix a demander.

Chapitre 13: Troisième cas - Product Pack

13.1. Domain engineering

13.1.1. Recherche de modules dans la communauté

En recherchant des modules dans la communauté nous trouvons un module intéressant. Ce module intitulé « product pack » permet de créer des packs de produits et ensuite de les revendre. Nous trouvons cette fonctionnalité intéressante pour de futur projet et nous décidons donc de l'incorporer à notre feature model.

13.1.2. Mise à jour du feature model et des schémas BPMN

Le feature model et le schéma mis à jour se trouvent dans l'annexe 8.

13.2. Application engineering

13.2.1. Analyse du besoin

Un commerçant désire pouvoir vendre des packs de produits, par exemple un pack ratatouille qui contiendrait des tomates, aubergines, poivrons, … à un prix préférentiel. Il dispose aussi de palettes de cannettes de boissons qu'il vend soit à l'unité ou par palettes de 22 canettes plus 2 gratuites.

13.2.2. Sélection d'une configuration

Suite à la lecture de l'analyse du besoin nous décidons d'installer le module « Sales » et le nouveau module « product pack ».

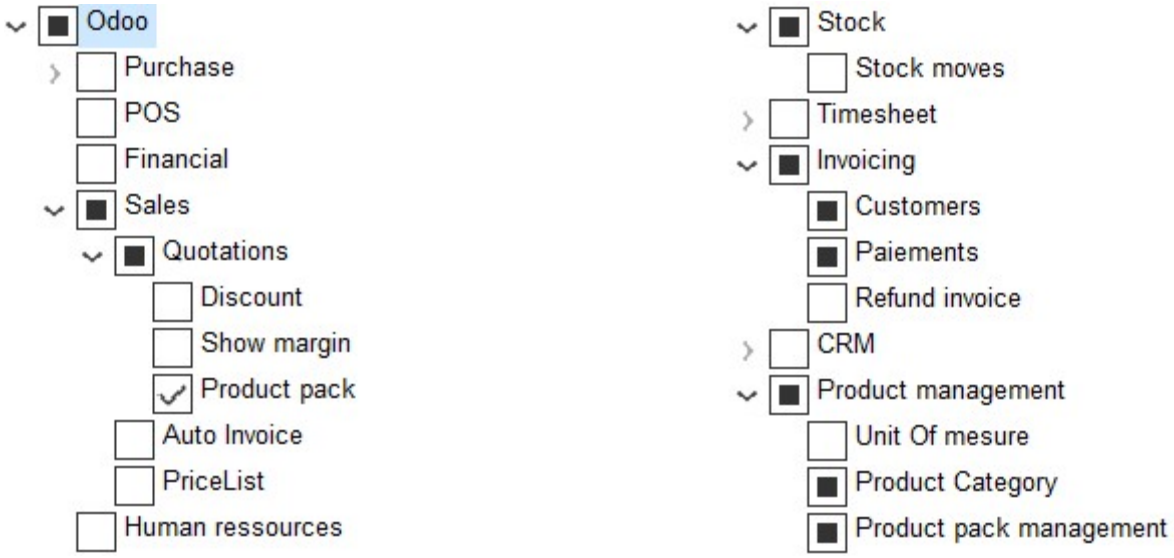


Figure 34 Configuration exemple 3

13.2.3. Test « Est-ce que tous les besoins sont couverts » ?

« Sur base de l'analyse des besoins, est-ce que la modélisation BPMN de la configuration sélectionnée couvre tous les besoins ? »

Oui, Il n'y a donc pas de développement à faire mais ce module doit être acheté. L'étape prototypage n'est donc pas nécessaire

13.2.4. Contractualisation

Nous savons maintenant que nous devons juste installer des modules et ne pas nous lancer dans des modifications. Cependant le module « product pack » nous coutera 69€ d'achat.

Partie V
CONCLUSION

Chapitre 14: Conclusion

Nous allons maintenant conclure par un bref récapitulatif de ce mémoire.

Nous avons commencé par notre état de l'art, dans lequel l'ERP Odoo a été présenté, ensuite nous vous avons présenté différentes méthodologies de mise en œuvre existantes. Nous avons ensuite brièvement présenté le « Software Product Line Engineering » et fait un rapprochement avec les ERPs. Après nous avons présenté notre outil de modélisation BPNN. Nous avons terminé en vous présentant la méthode de mesure par points de fonctions et son application à un cas avec un ERP et un autre cas BPMN.

Un des objectifs de ce mémoire était d'apporter une méthodologie d'implémentation pour l'ERP Odoo simple et adaptée pour les PME c'est ce que nous avons fait dans notre contribution après avoir expliqué les différentes modifications de Odoo possibles et critiquer les méthodologies vues dans l'état de l'art. Le deuxième objectif était de proposer une modélisation de Odoo ce que nous avons fait au moyen d'un feature model et des diagrammes BPMN dans le chapitre 9. Enfin notre dernier objectif était de proposer un moyen d'estimer la taille des modifications éventuelles apportées à Odoo au moyen de Cosmic ce qui a été le sujet du chapitre 10.

Dans la dernière partie avant cette conclusion, nous avons appliqué notre méthodologie à trois cas pratique.

La force de ce mémoire consiste en la recherche de la réutilisation maximale des fonctionnalités déjà existantes ce qui permet d'éviter des frais de développements.

Une faiblesse de ce mémoire serait la transition d'une configuration d'un feature model au diagramme BPMN et la précision des digramme BPMN.

14.1. Perspectives d'évolution

Pour finir nous allons vous présenter quelques perspectives d'évolution possibles.

La première amélioration serait de tester encore plus la méthodologie dans d'autres projet afin de connaitre sa pertinence et éventuellement savoir comment on pourrait encore l'améliorer.

La seconde amélioration intéressante serait un outil logiciel qui permettrait de construire automatiquement un diagramme BPMN au moyen d'une configuration du feature model donné.

Finalement, on pourrait aussi rajouter les étapes de quantification de la taille de l'installation, de l'importation des données et de la paramétrisation de l'ERP dans la méthodologie qui n'ont pas été abordées dans ce mémoire.

Partie VI
SOURCES

Bibliographie

Anquetil, Nicolas, et al. 2008. *Lignes de produits logiciels et usines.* 2008.

Ben Rhouma, Takoua. 2012. *Composition des modèles de lignes de produits logiciels.* Paris XI : Université Paris Sud, 2012. tel-00772257.

Beuche, Danilo et Dalgarno, Mark. 2007. Software Product Line Engineering with Feature Models. *www.pure-systems.com.* [En ligne] 2007. https://www.pure-systems.com/mediapool/tutorial_splwithfeaturemodelling.pdf.

Boutin, P. 2001. Définition d'une méthodologie de mise en oeuvre et de prototypage d'un progiciel de gestion d'entreprise (ERP). s.l. : Ecole Nationale Supérieure des Mines de Saint-Etienne, 2001. HAL Id: tel-00850069.

Cosmic. 2016. *Introduction à la méthode de mesurage des logiciels COSMIC.* 2016.

—. 2015. *Manuel de mesurage v4.0.1.* 2015.

Lacombe, Clément. 2015. *Contribution à une méthodologie et une modélisation pour accompagner les petites entreprises dans l'étude de leur organisation afin de spécifier leurs besoins et sélectionner une solution ERP.* s.l. : Université de Bordeaux, 2015. <NNT : 2015BORD0430>, <tel-01282022>.

Lucidchart. What is Business Process Modeling Notation. *www.lucidchart.com.* [En ligne] [Citation : 2017 08 06.] https://www.lucidchart.com/pages/bpmn .

lucidchart, BPMN Gateway Types. BPMN Gateway Types. *www.lucidchart.com.* [En ligne] [Citation : 06 08 2017.] https://www.lucidchart.com/pages/bpmn/gateways.

Marín, Beatriz et Quinteros, José. 2012. *A COSMIC Measurement Procedure for BPMN Diagrams.* s.l. : Facultad de Ingeniería Universidad Diego Portales, 2012. (Ref. 11121395, 2012-2015).

Montero, Ildefonso, Peña, Joaquín et Ruiz-Cortés, Antonio. 2008. From Feature Models to Business Processes. [En ligne] 08 2008. http://yourtask.webcindario.com/yourtask/doc/montero-papers/montero-scc08.pdf. DOI: 10.1109/SCC.2008.130.

Mordant, Alain . 2007. A model-based methodology for early stages of ERP implementations in SMEs. s.l. : FUNDP, 2007.

MOTAKI, Noureddine , KAMACH , Oulaid et DERBOUL, Ahmed. 2015. *Approche Processus pour l'intégration d'un système.* Tanger - Maroc. : s.n., 2015.

Object Management Group. 2013. Business Process Model and Notation. *omg.org.* [En ligne] 10 12 2013. [Citation : 06 08 2017.] http://www.omg.org/cgi-bin/doc?formal/13-12-10.pdf.

Object Management Group. 2010. BPMN 2.0 by Example. *omg.org.* [En ligne] 02 06 2010. [Citation : 06 08 2017.] http://www.omg.org/cgi-bin/doc?dtc/10-06-02.odt. dtc/2010-06-02.

Odoo, Actions. Actions. [En ligne] https://www.odoo.com/documentation/8.0/reference/actions.html.

Odoo, Qweb. Qweb. [En ligne] https://www.odoo.com/documentation/8.0/reference/qweb.html#reference-qweb.

Odoo, Views. Views. [En ligne]
https://www.odoo.com/documentation/8.0/reference/views.html.

Odoo, Workflow. Workflow. [En ligne]
https://www.odoo.com/documentation/8.0/reference/workflows.html.

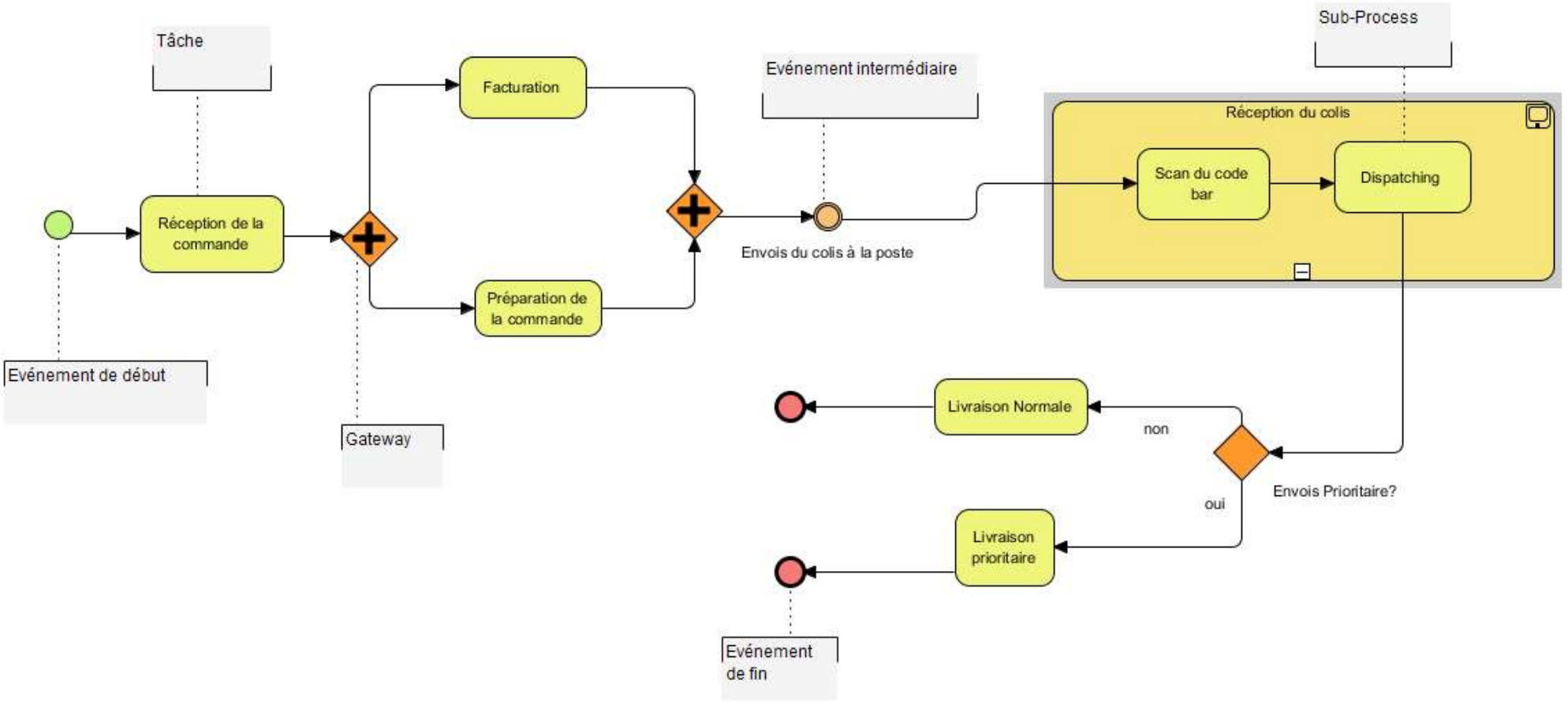
Odoo,How To Backend. How To Backend. [En ligne] [Citation : 06 08 2017.]
https://www.odoo.com/documentation/8.0/howtos/backend.html.

Schnieders, Arnd et Puhlmann, Frank. 2006. Variability Mechanisms in E-Business Process. [En ligne] 2006. http://www.frapu.de/pdf/BIS2006-PESOA.pdf.

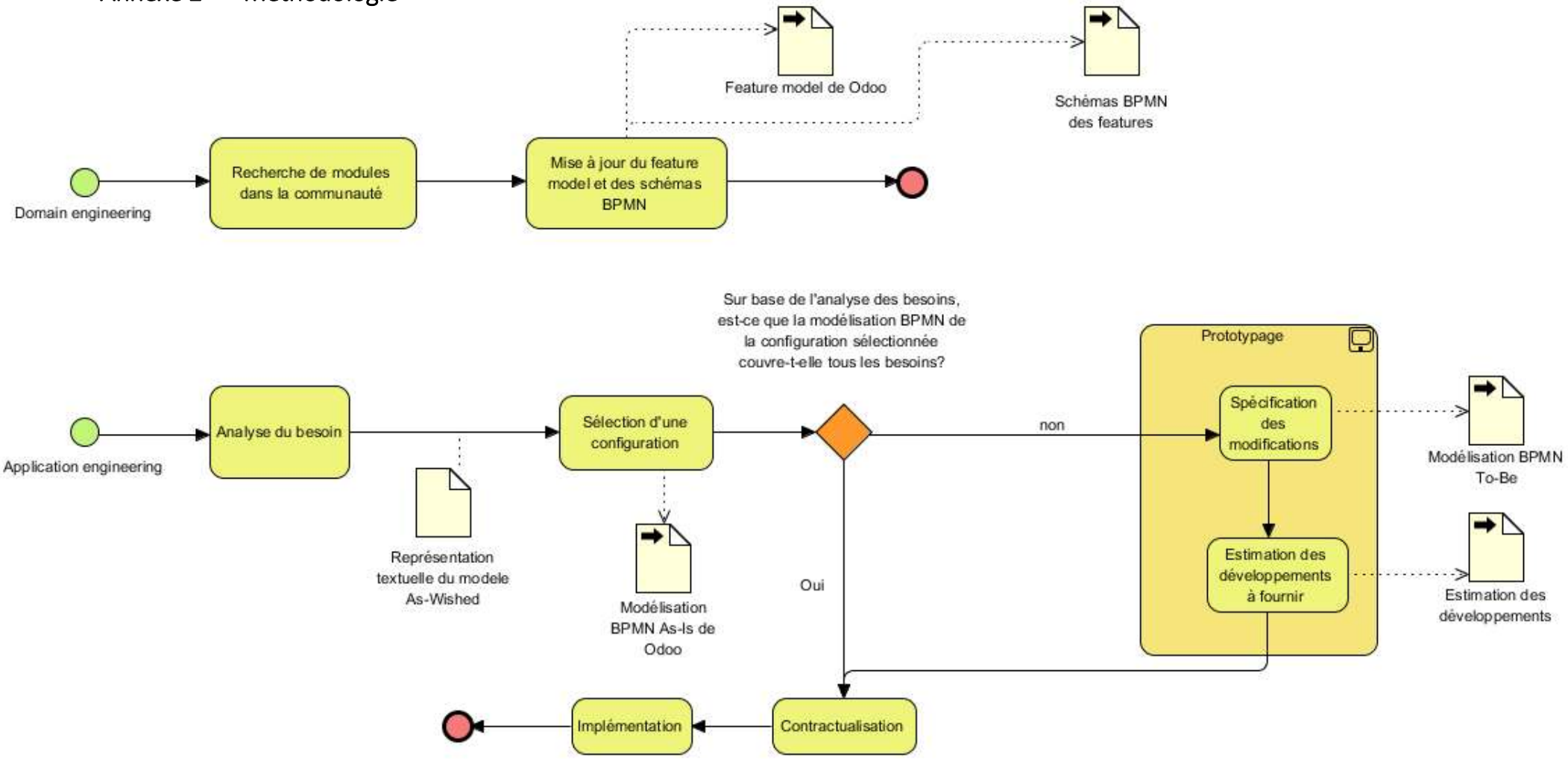
Téllez, Francisco Martín. 2009. *Solving the size estimation problem in ERP project context: the eEPC-COSMIC approach.* 2009.

Partie VII
ANNEXES

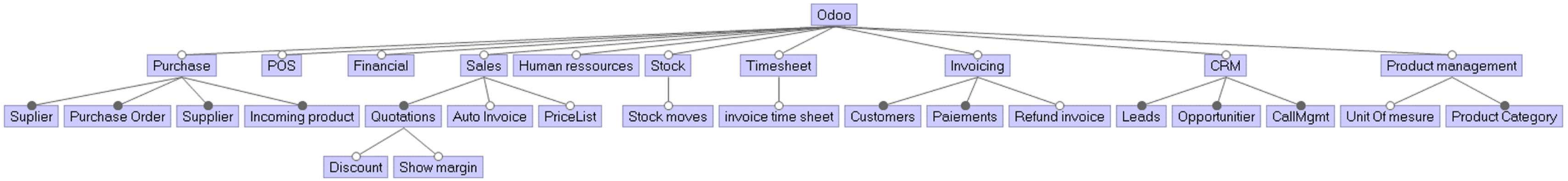
Annexe 1 Exemple BPMN



Annexe 2 Méthodologie



Annexe 3 Feature model de Odoo

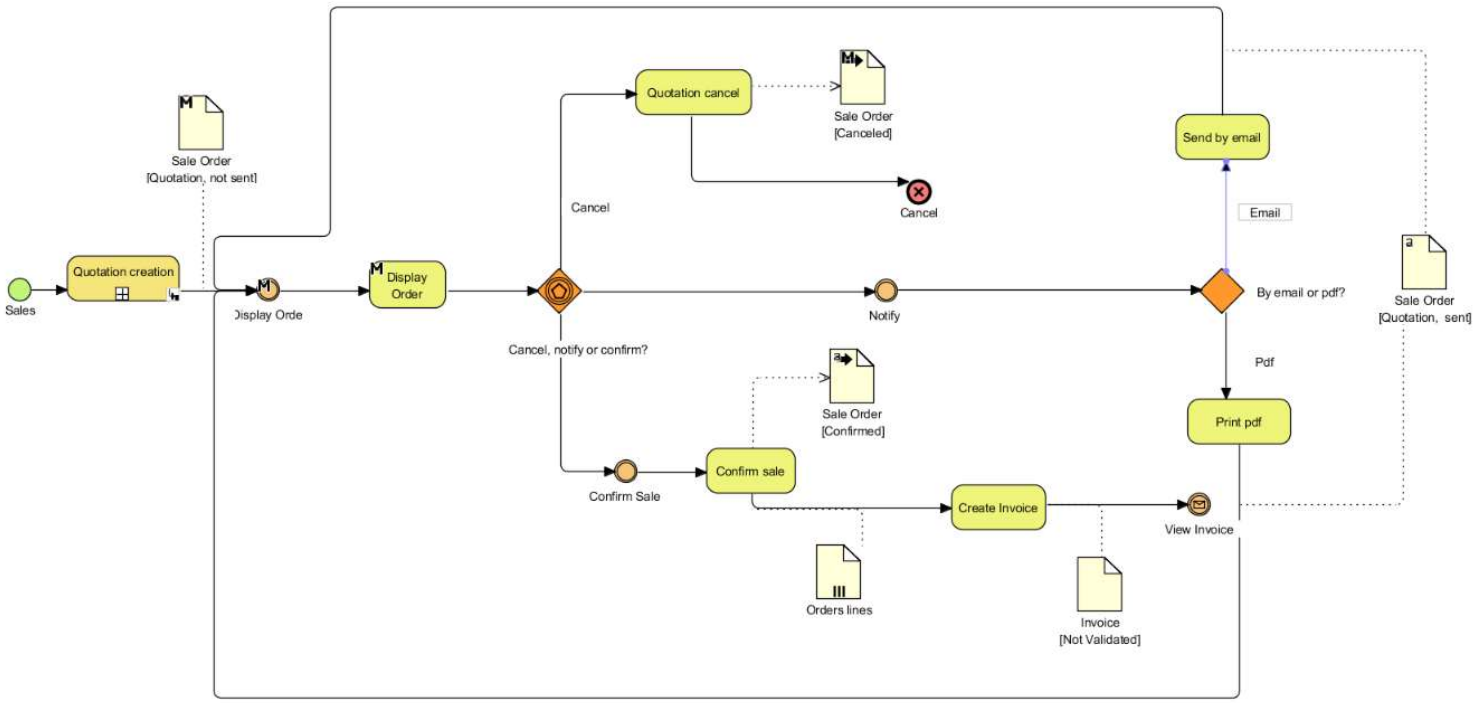


Timesheet ⇒ "Human ressources"
Timesheet ⇒ Invoicing
"invoice time sheet" ⇒ Sales
Purchase ⇒ Invoicing
Purchase ⇒ Stock
POS ⇒ Stock
POS ⇒ Sales
Financial ⇒ Invoicing
Sales ⇒ "Product management"
Sales ⇒ Invoicing

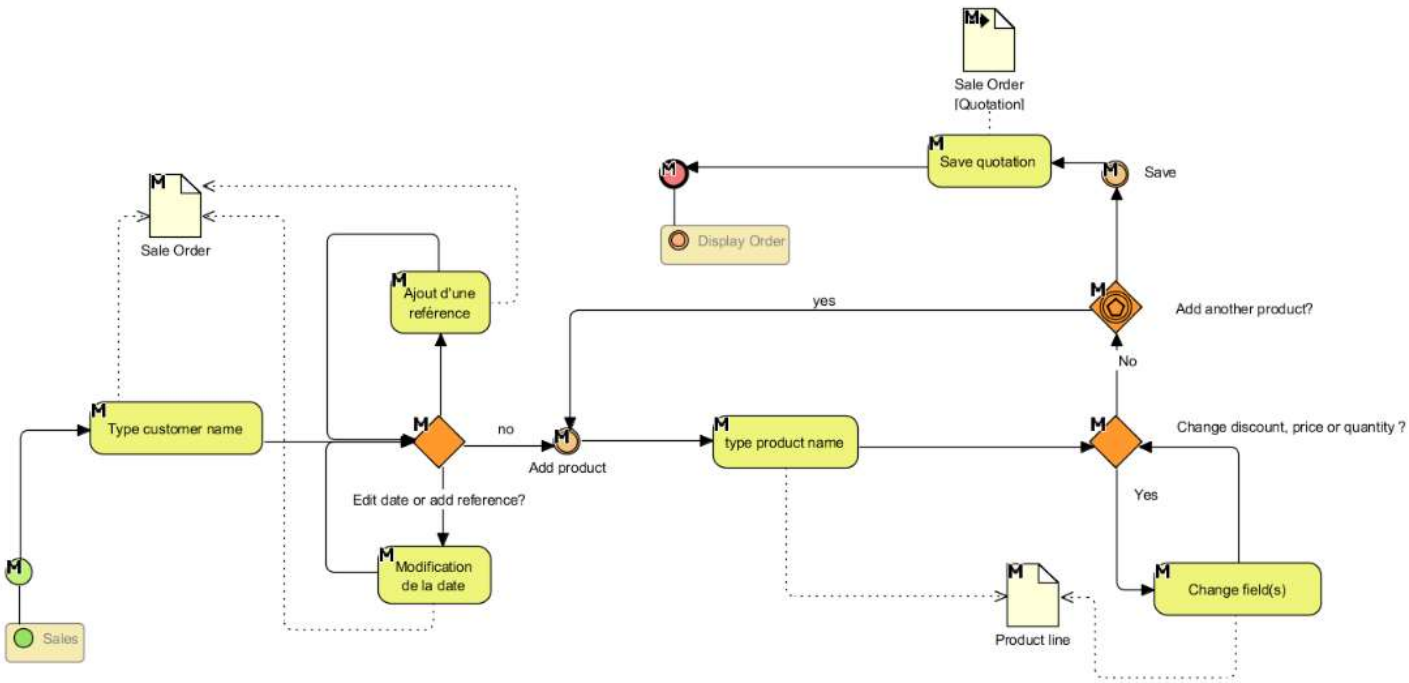
Legend:
Mandatory
Optional

Annexe 4 Modélisation BPMN de Odoo

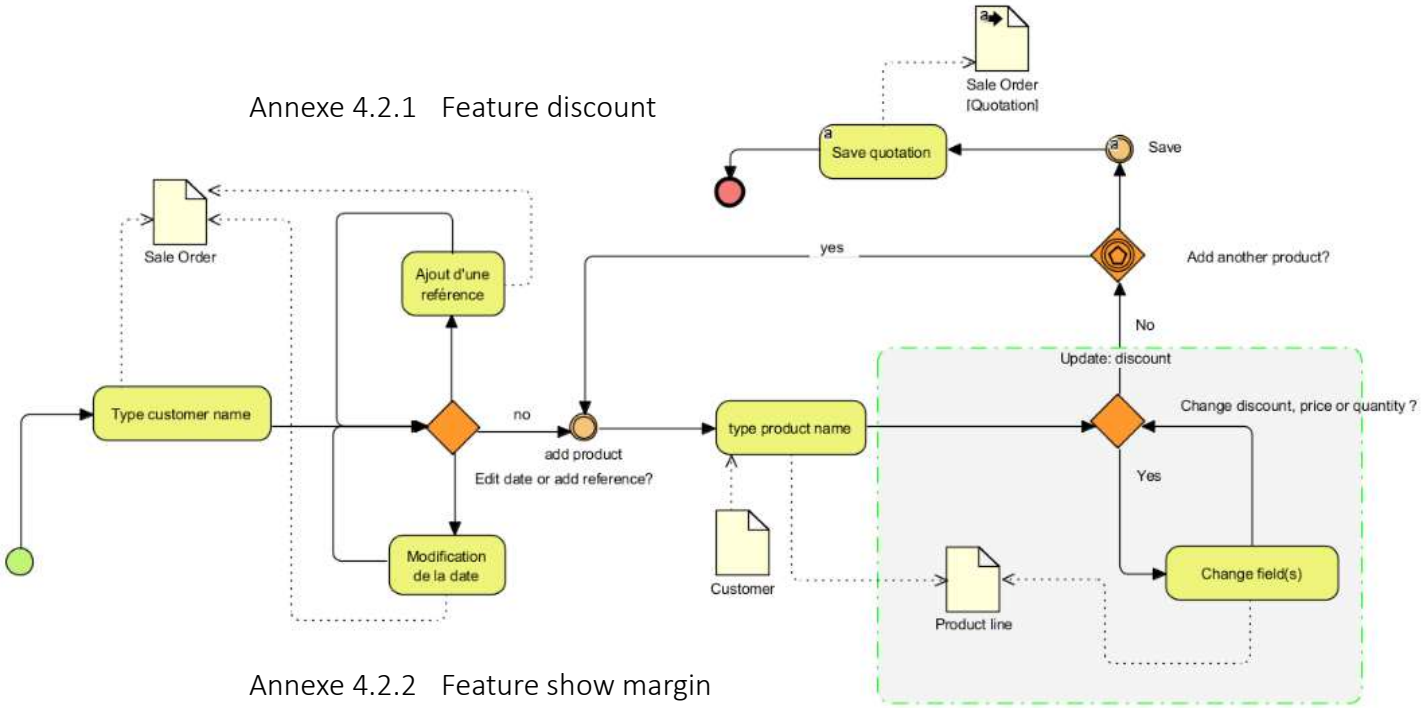
Annexe 4.1 Feature Sales



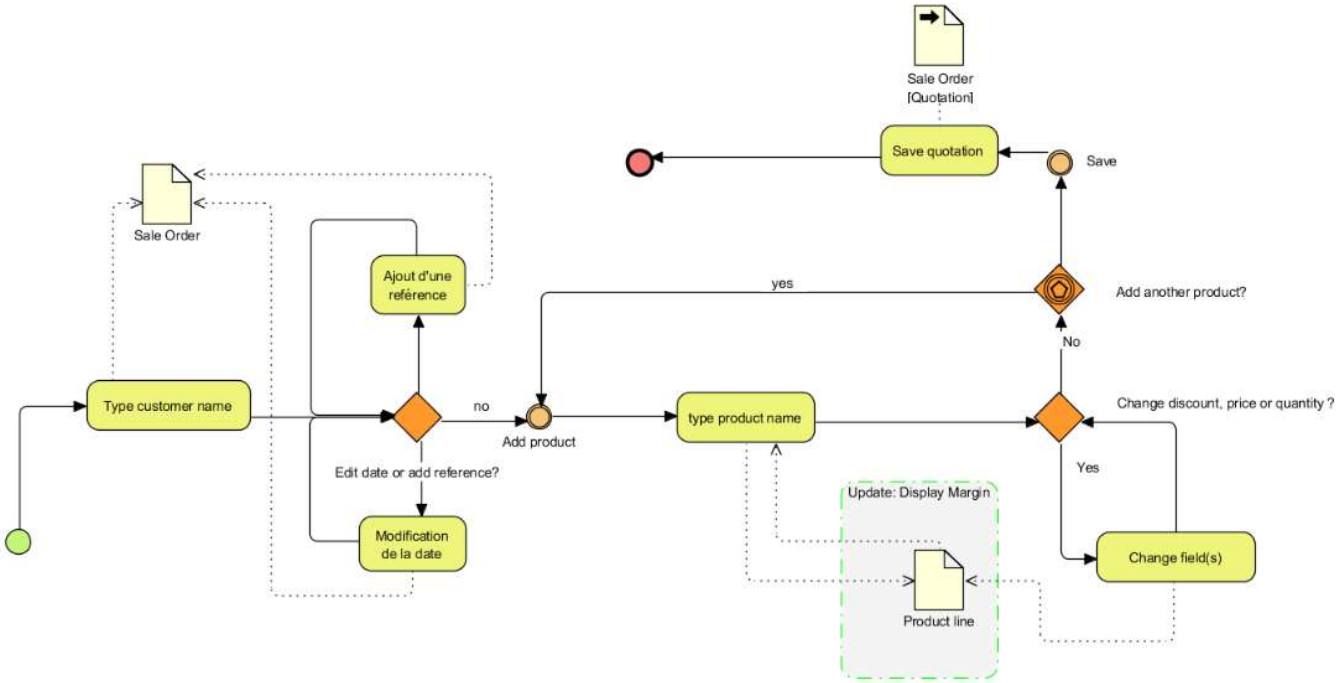
Annexe 4.2 Feature Quotation



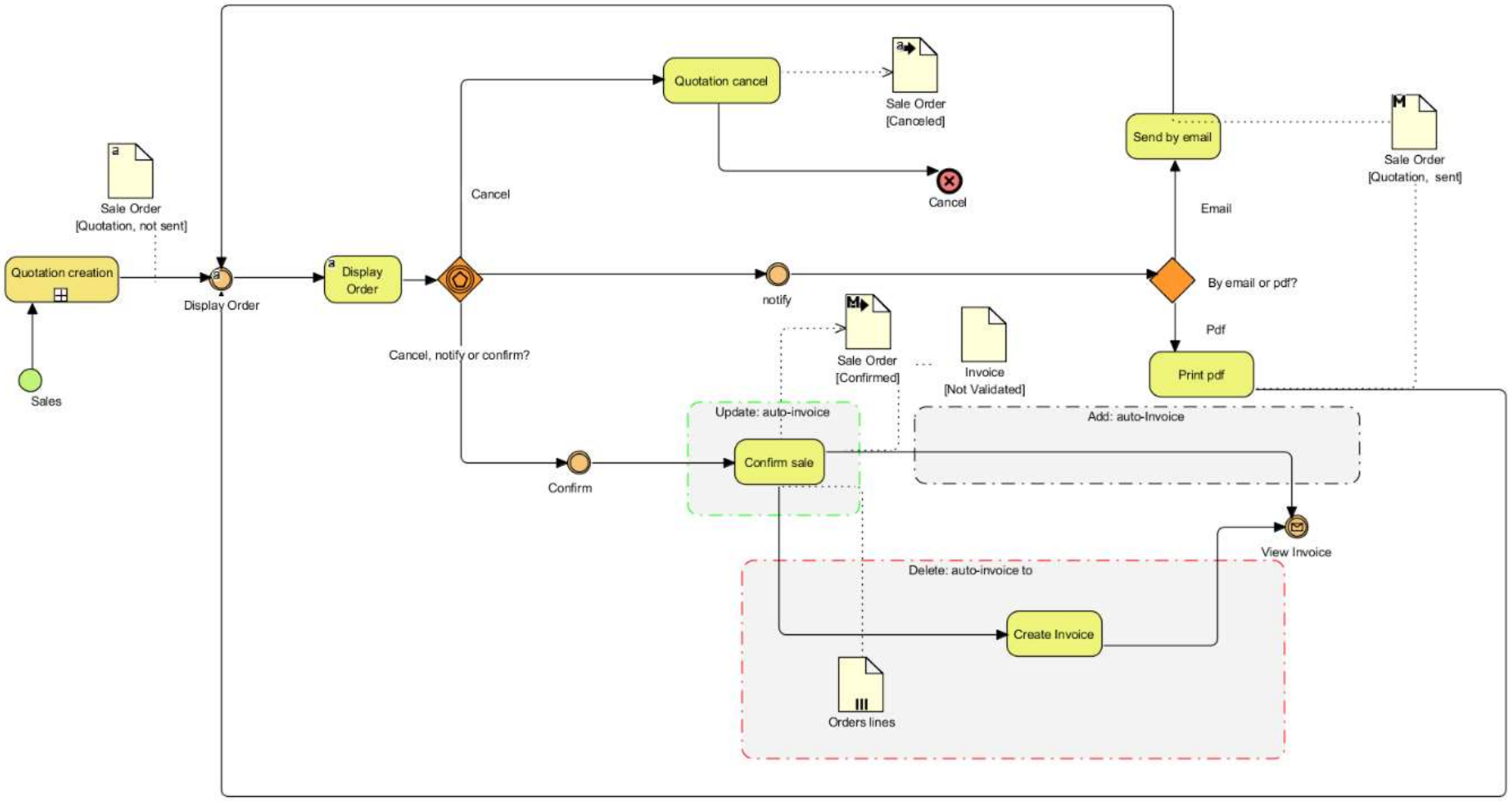
Annexe 4.2.1 Feature discount



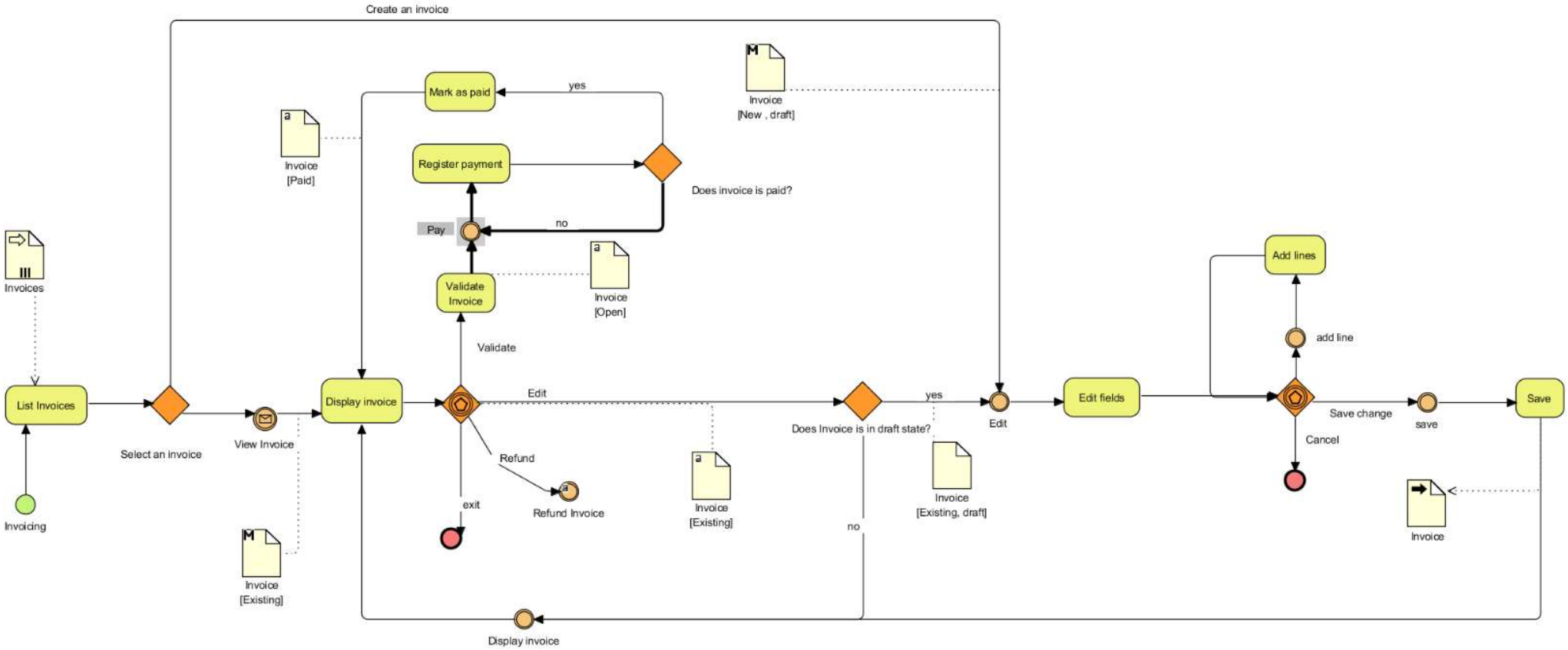
Annexe 4.2.2 Feature show margin



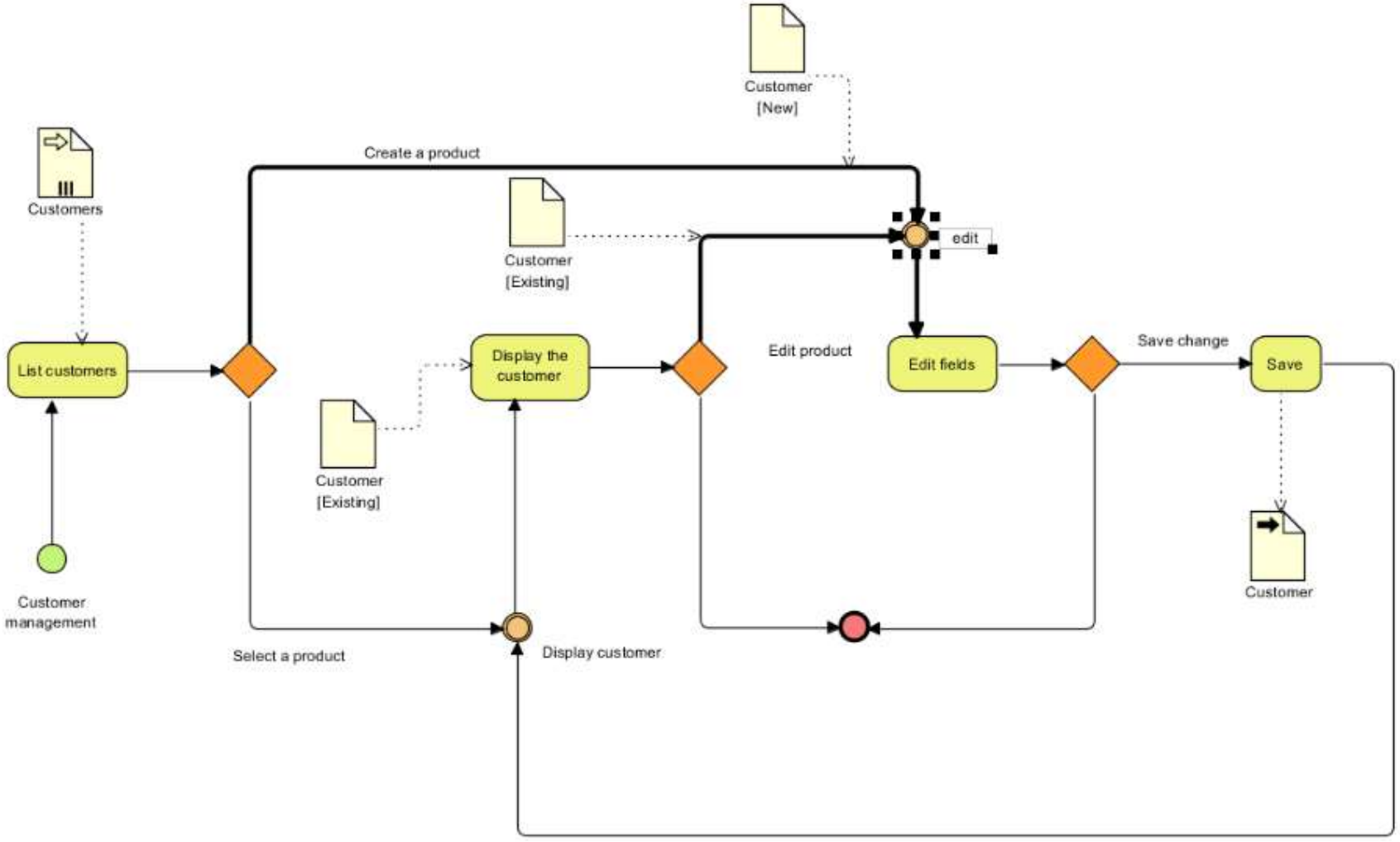
Annexe 4.3 Feature Auto invoice



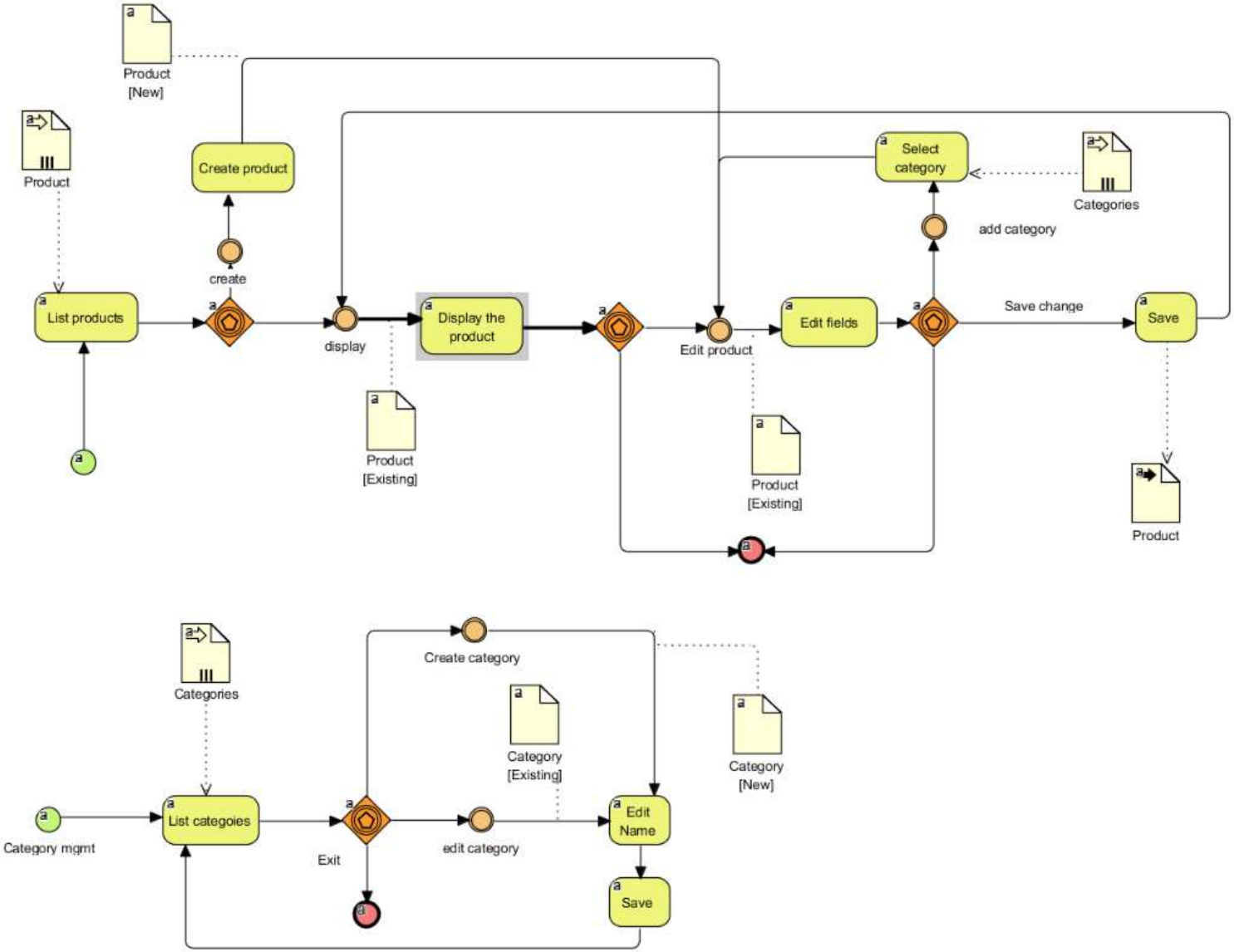
Annexe 4.4 Invoicing



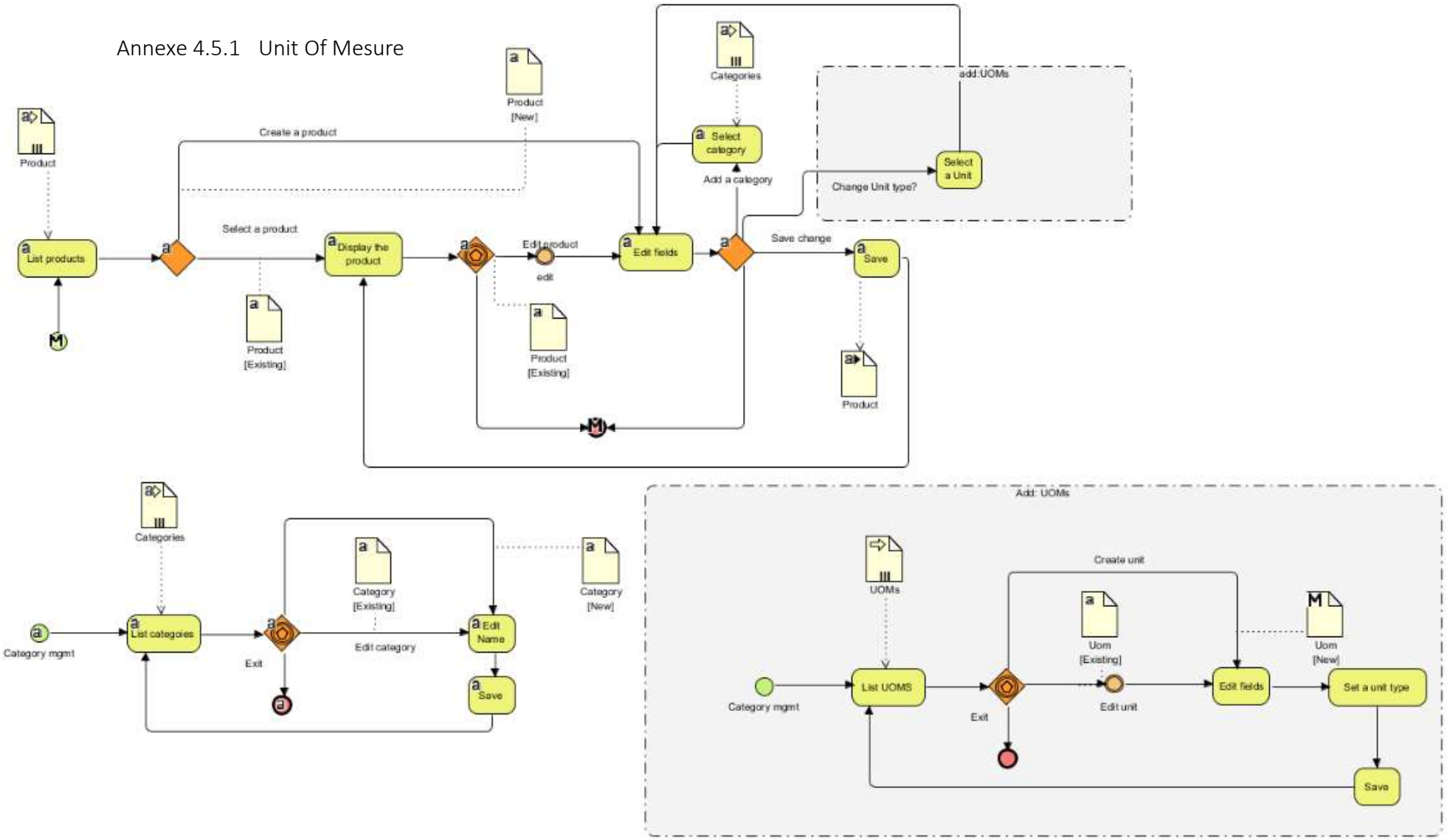
Annexe 4.4.1 Customer



Annexe 4.5 Product Management



Annexe 4.5.1 Unit Of Mesure



Annexe 5 Description des data Object

Annexe 5.1 Sale Order

Name	Type	Default	Mandatory?
State	State	Draft Quotation	yes
Date	Date	Now	yes
isPaid	Boolean	false	yes
Payment term	Payment Term		
Fiscal Position	Fiscal Position		
SalesPerson	Customer		
Order Lines	Multi Order Lines		yes

States: Draft Quotation, Quotation Sent, Sales Order, Sale to Invoice, Done

Annexe 5.2 Order Line

Name	Type	Default	Mandatory?
productName	text	copied from Product	yes
description	text	copied from Product	yes
Quatity	integer	1	yes
price	float	copied from Product	yes
Taxes	Multi Taxes		

Annexe 5.3 Invoice

Name	Type	Default	Mandatory?
Customer	Partner		
Fiscal Position	Fiscal Position	from partner	
date	Date	now	
lines	Innvoice line		
Salesperson			
Bank Account			
Due date			
Payments	Multi Payment		
Sate			

States : Darft, Open, Paid

Annexe 5.4 Invoice line

Name	Type	Default	Mandatory?
productName	text	copied from Product	yes
description	text	copied from Product	yes
Quatity	integer	1	yes
price	float	copied from Product	yes
Taxes	Multi Taxes		

Annexe 5.5 Payment

Name	Type	Default	Mandatory?
Amount	float	Invoice amount	yes
Date	Date	Now	no
period	text	current period	yes
payment method	text		yes

Annexe 5.6 Customer

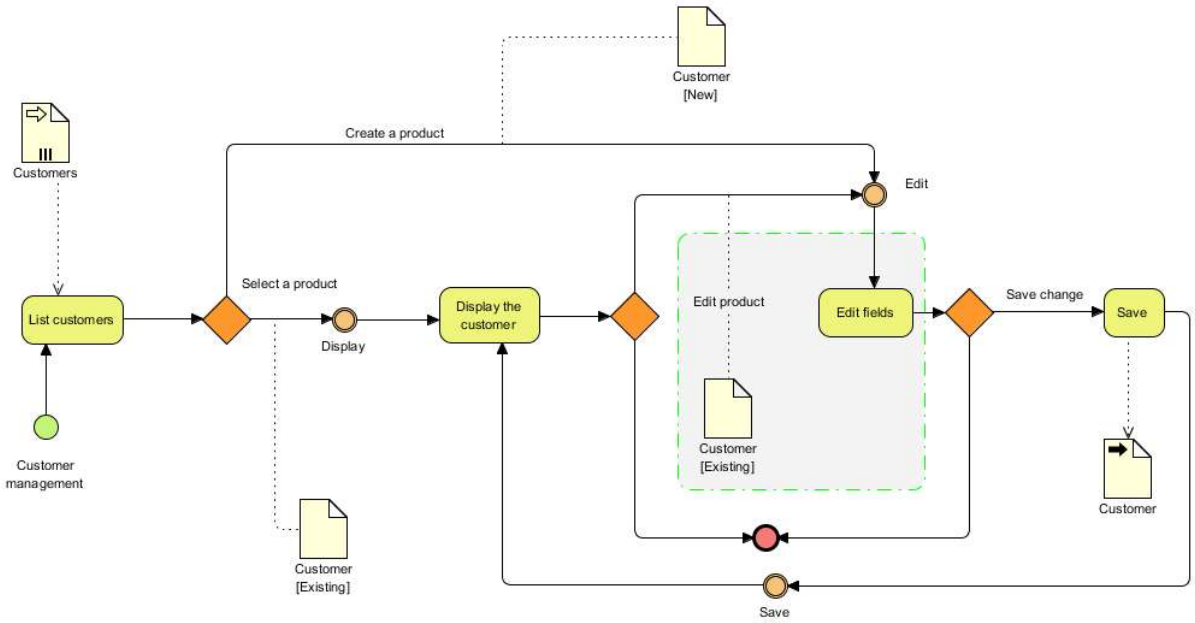
Name	Type	Default	Mandatory?
Name	Text		yes
isCompany	boolean		
address	text		
Phone	text		
Title	text		
…			

Annexe 5.7 Product

Name	Type	Default	Mandatory?
Name	Text		yes
Product Type	Product Type	Consumable	yes
Sale Price			
isActive	boolean	true	yes
ean13			
Reference			
Description			
Cost Price			
Manager	Partner		
Volume	float		
Gross Weight	float		
Net Weight	float		
Waranty	float (months)	0	
Catégory	Category		
Customer Taxes	Multi Taxe		
Supplier Taxes			

Annexe 6 Modification pour l'exemple n°1

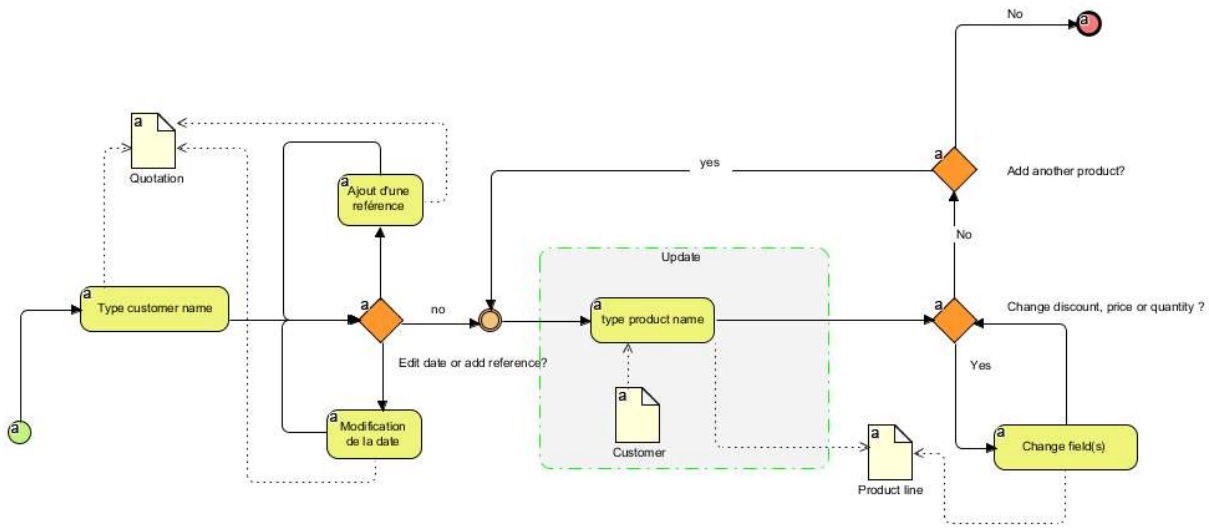
Annexe 6.1 Modification du diagramme BPMN du client



Annexe 6.2 Data object Customer modifié

Name	Type	Default	Mandatory?
Name	Text		yes
isCompany	boolean		
address	text		
Phone	text		
Title	text		
Discount	Float	0	yes

Annexe 6.3 Modification du diagramme BPMN du devis

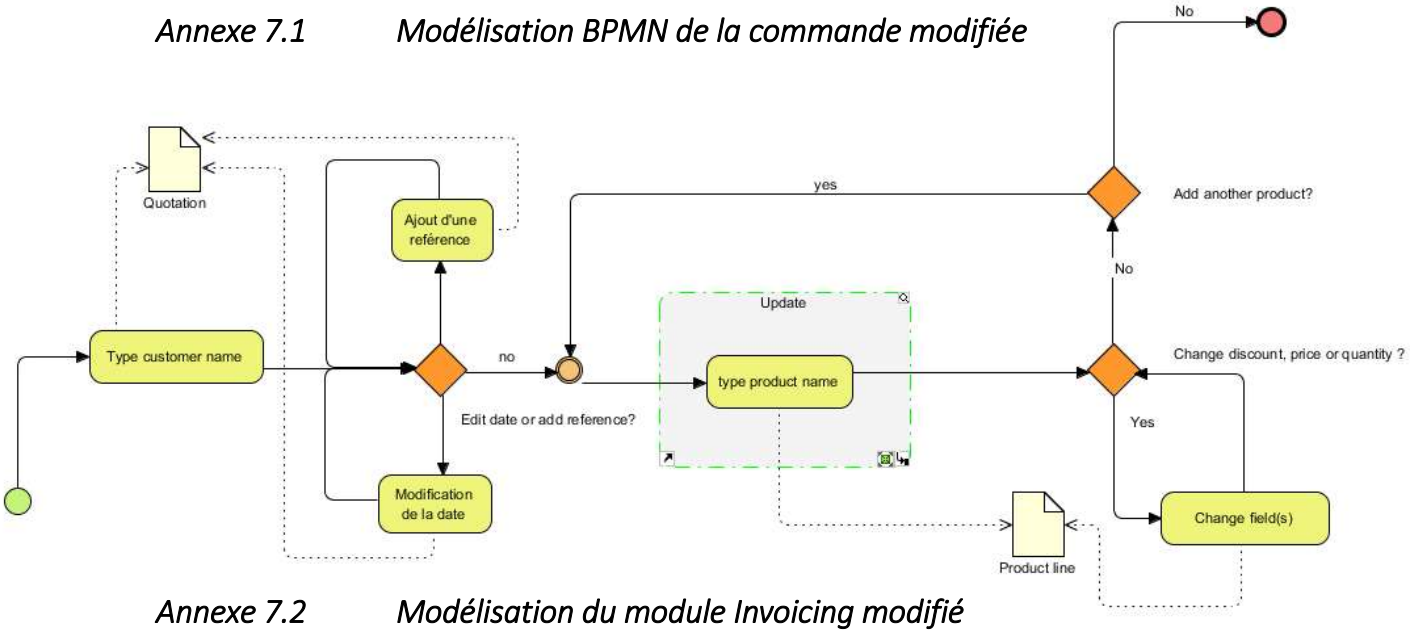


Annexe 6.4 Estimation des dévlopement

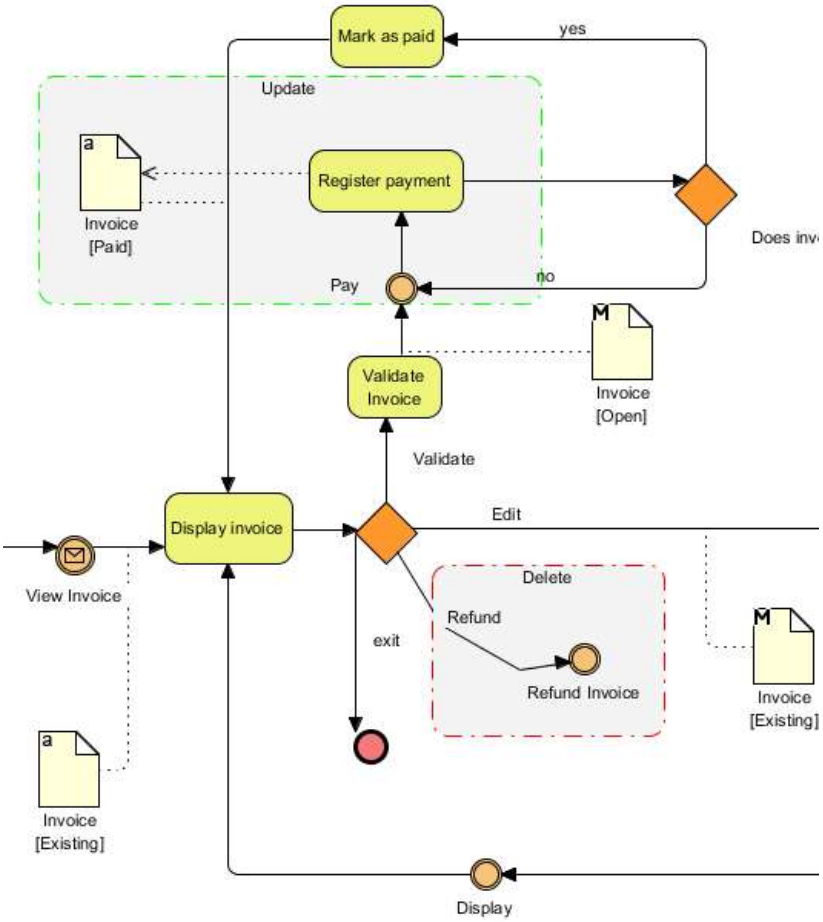
Module	FUR	Entrée	Sortie	Lecture	éCriture	Total
Customer	Edit fields	1			1	2
Quotation	Type product Name	1		1		2
	Totaux	2	0	1	1	4

Annexe 7 Modification pour l'exemple n°2

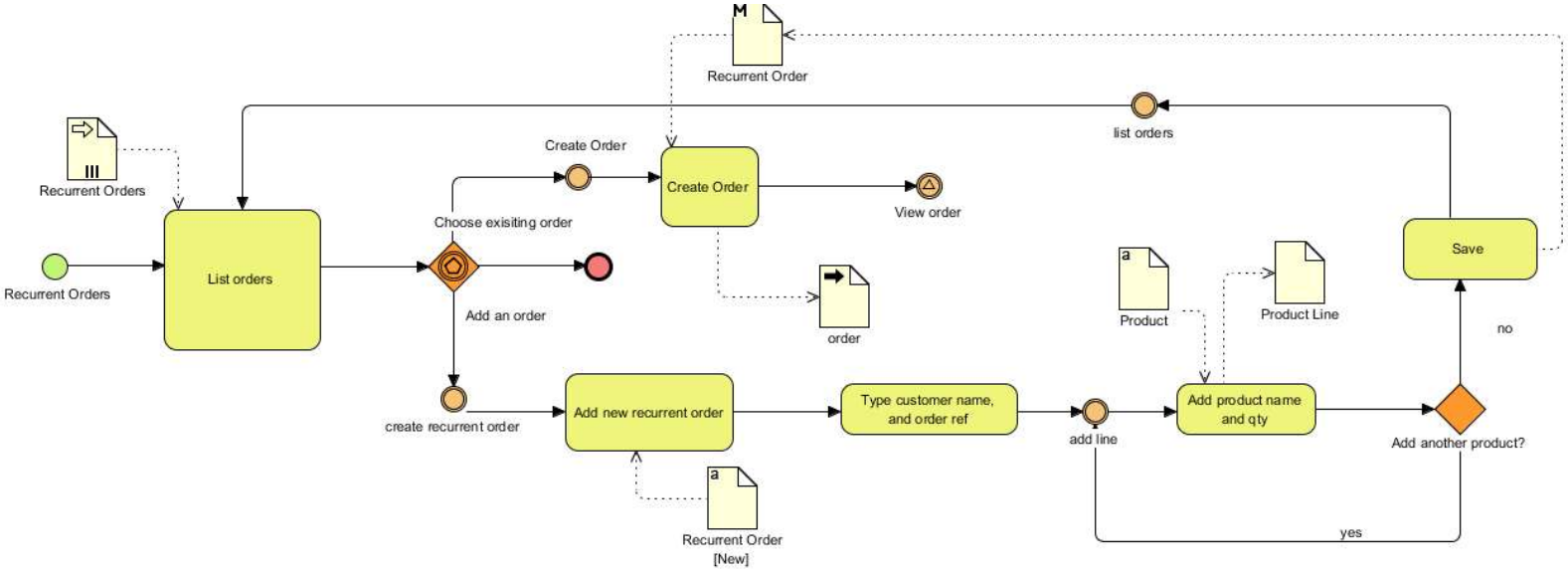
Annexe 7.1 Modélisation BPMN de la commande modifiée



Annexe 7.2 Modélisation du module Invoicing modifié



Annexe 7.3 Modélisation du nouveau module commande récurrente

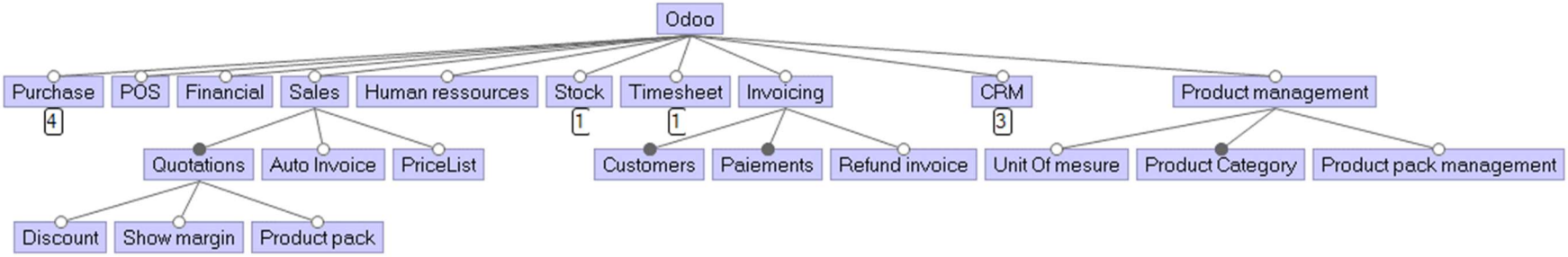


Annexe 7.4 Estimation

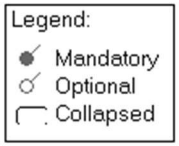
Module	FUR	Entrée	Sortie	Lecture	éCriture	Total	Total/module
Quotation	Type product name	1				1	1
Invoicing	Register payment	1		1		2	
	Refund		1			1	3
Custom	List order	1	3	1		5	
	Create order	1	1	1	1	4	
	Add new order	1		1		2	
	Types ..	1				1	
	add product name and qty	1	1	1	1	4	
	Save	1	1		1	3	19
	Totaux	8	7	5	3	23	

Annexe 8 Exemple n°3

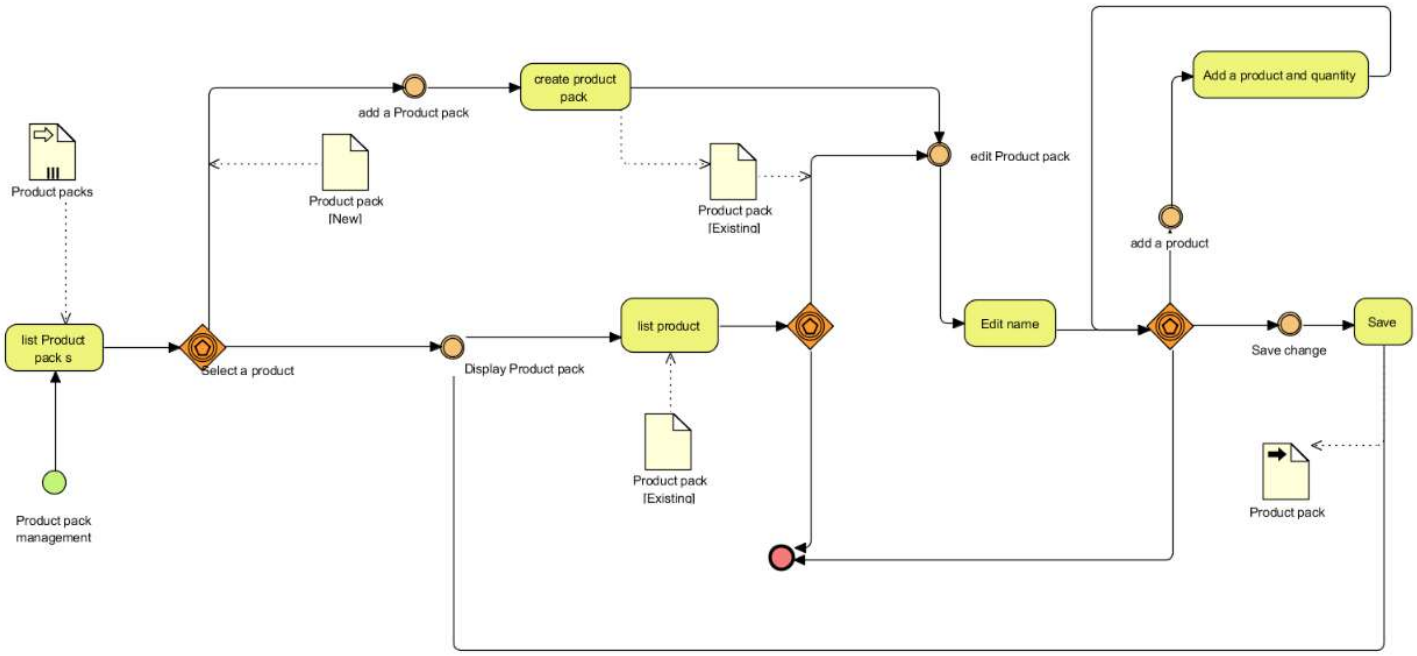
Annexe 8.1 Nouveau feature model



Timesheet ⇒ "Human ressources"
Timesheet ⇒ Invoicing
"invoice time sheet" ⇒ Sales
Purchase ⇒ Invoicing
Purchase ⇒ Stock
POS ⇒ Stock
POS ⇒ Sales
Financial ⇒ Invoicing
Sales ⇒ "Product management"
Sales ⇒ Invoicing
"Product pack management" ⇒ Stock
"Product pack" ⇒ "Product pack management"



Annexe 8.2 Modélisation BPMN Product pack



Annexe 8.3 Modification